Chalan-Pro Source Code Deposit

Author: Oliver Alberto Hernandez Perez
Subject: Copyright Office Deposit - Chalan-Pro
Date: August 24, 2025

This PDF aggregates selected source files for copyright deposit.

Table of Contents

File path → starting page

Table of Contents (cont.)

Table of Contents (cont.)

appinventory__init__.py

```
00001:
```

appinventory\admin.py

```
00001: from django.contrib import admin
00002: from .models import (
00003:     UnitCategory, UnitOfMeasure, Warehouse,
00004:     ProductCategory, ProductBrand, Product,
00005:     PriceType, ProductPrice,
00006:     Stock, InventoryMovement
00007: )
00008:
00009: @admin.register(UnitCategory)
00010: class UnitCategoryAdmin(admin.ModelAdmin):
00011:     list_display = ('name', 'description', 'is_active')
00012:     search_fields = ('name',)
00013:
00014:
00015: @admin.register(UnitOfMeasure)
00016: class UnitOfMeasureAdmin(admin.ModelAdmin):
00017:     list_display = ('code', 'name', 'category', 'reference_unit', 'conversion_sign', '
conversion_factor', 'is_active')
00018:     list_filter = ('category', 'reference_unit', 'is_active')
00019:     search_fields = ('name', 'code')
00020:     autocomplete_fields = ['category']
00021:
00022:
00023: @admin.register(Warehouse)
00024: class WarehouseAdmin(admin.ModelAdmin):
00025:     list_display = ('name', 'location', 'is_active')
00026:     search_fields = ('name', 'location')
00027:
00028:
00029: @admin.register(ProductCategory)
00030: class ProductCategoryAdmin(admin.ModelAdmin):
00031:     list_display = ('name', 'is_active')
00032:     search_fields = ('name',)
00033:
00034:
00035: @admin.register(ProductBrand)
00036: class ProductBrandAdmin(admin.ModelAdmin):
00037:     list_display = ('name', 'is_active')
00038:     search_fields = ('name',)
00039:
00040:
00041: class ProductPriceInline(admin.TabularInline):
00042:     model = ProductPrice
00043:     extra = 1
00044:     autocomplete_fields = ['unit', 'price_type']
00045:
00046:
00047: @admin.register(Product)
00048: class ProductAdmin(admin.ModelAdmin):
00049:     list_display = ('name', 'sku', 'category', 'brand', 'unit_default', 'is_active')
00050:     list_filter = ('category', 'brand', 'is_active')
00051:     search_fields = ('name', 'sku')
00052:     autocomplete_fields = ['category', 'brand', 'unit_default']
00053:     inlines = [ProductPriceInline]
00054:
00055:
00056: @admin.register(PriceType)
00057: class PriceTypeAdmin(admin.ModelAdmin):
00058:     list_display = ('name', 'description', 'is_active')
00059:     search_fields = ('name',)
```

appinventory\admin.py

```
00060:
00061:
00062: @admin.register(Stock)
00063: class StockAdmin(admin.ModelAdmin):
00064:     list_display = ('product', 'warehouse', 'quantity')
00065:     list_filter = ('warehouse',)
00066:     search_fields = ('product__name', 'warehouse__name')
00067:     autocomplete_fields = ['product', 'warehouse']
00068:
00069:
00070: @admin.register(InventoryMovement)
00071: class InventoryMovementAdmin(admin.ModelAdmin):
00072:     list_display = ('product', 'warehouse', 'quantity', 'movement_type', 'unit', 'docu
ment', 'timestamp')
00073:     list_filter = ('movement_type', 'timestamp', 'warehouse')
00074:     search_fields = ('product__name', 'document')
00075:     autocomplete_fields = ['product', 'warehouse', 'unit', 'created_by']
```

appinventory\apps.py

```
00001: from django.apps import AppConfig
00002:
00003:
00004: class AppinventoryConfig(AppConfig):
00005:     default_auto_field = 'django.db.models.BigAutoField'
00006:     name = 'appinventory'
```

appinventory\helpers.py

```
00001: from decimal import Decimal, ROUND_HALF_UP, InvalidOperation
00002:
00003: def convert_to_reference_unit(product, unit, quantity):
00004:     print("■ appinventory-helpers.py : This is convert_to_reference_unit")
00005:     """
00006:     Convierte una cantidad a la unidad de referencia del producto.
00007:     Si la conversión falla, devuelve la cantidad original o 0 como fallback.
00008:     """
00009:     if quantity is None:
00010:         print("■ ERROR: convert_to_reference_unit recibió quantity=None")
00011:         return Decimal('0.00')
00012:
00013:     try:
00014:         quantity = Decimal(quantity)
00015:     except (InvalidOperation, TypeError) as e:
00016:         print(f"■ ERROR al convertir quantity a Decimal: {e}")
00017:         return Decimal('0.00')
00018:
00019:     if unit == product.unit_default:
00020:         return quantity.quantize(Decimal('0.01'), rounding=ROUND_HALF_UP)
00021:
00022:     factor = unit.conversion_factor
00023:     sign = unit.conversion_sign
00024:
00025:     if not factor or factor <= 0:
00026:         print("■■ Factor inválido, devolviendo quantity sin conversión.")
00027:         return quantity.quantize(Decimal('0.01'), rounding=ROUND_HALF_UP)
00028:
00029:     try:
00030:         if sign == '*':
00031:             result = quantity * factor
00032:         elif sign == '/':
00033:             result = quantity / factor
00034:         else:
00035:             result = quantity
00036:         return Decimal(result).quantize(Decimal('0.01'), rounding=ROUND_HALF_UP)
00037:     except Exception as e:
00038:         print(f"■ Error en conversión matemática: {e}")
00039:         return quantity.quantize(Decimal('0.01'), rounding=ROUND_HALF_UP)
```

appinventory\management\commands__init__.py

```
00001:
```

appinventory\management\commands\recalculate_stock.py

```
00001: from django.core.management.base import BaseCommand
00002: from apptransactions.models import DocumentLine
00003: from appinventory.models import Stock
00004: from django.db import transaction
00005: from appinventory.helpers import convert_to_reference_unit
00006:
00007: # python manage.py recalculate_stock
00008:
00009: class Command(BaseCommand):
00010:     help = 'Recalculate stock levels for all products and warehouses based on document
 lines.'
00011:
00012:     @transaction.atomic
00013:     def handle(self, *args, **kwargs):
00014:         self.stdout.write("■ Clearing current stock levels...")
00015:         Stock.objects.all().delete()
00016:
00017:         self.stdout.write("■ Recalculating stock from document lines...")
00018:         count = 0
00019:
00020:         lines = DocumentLine.objects.select_related(
00021:             'document', 'product', 'unit', 'warehouse', 'document__document_type'
00022:         )
00023:
00024:         for line in lines:
00025:             stock_movement = line.document.document_type.stock_movement
00026:             if not stock_movement:
00027:                 continue  # Neutral document type, skip
00028:
00029:             warehouse = line.warehouse or line.document.warehouse
00030:             if not warehouse:
00031:                 continue  # No warehouse defined
00032:
00033:             converted_qty = convert_to_reference_unit(line.product, line.unit, line.qu
antity)
00034:
00035:             stock, created = Stock.objects.get_or_create(
00036:                 product=line.product,
00037:                 warehouse=warehouse,
00038:                 defaults={'quantity': 0}
00039:             )
00040:             stock.quantity += converted_qty * stock_movement
00041:             stock.save()
00042:             count += 1
00043:
00044:             self.stdout.write(
00045:                 f"■ Product: {line.product.name} | Warehouse: {warehouse.name} | type:
 { stock_movement } | Qty: {converted_qty:.2f} -> Total: {stock.quantity:.2f}"
00046:             )
00047:
00048:         self.stdout.write(self.style.SUCCESS(f"■ Stock successfully recalculated from
{count} document lines."))
```

appinventory\management\commands\validate_unit_prices.py

```
00001: from django.core.management.base import BaseCommand
00002: from appinventory.models import ProductUnit, UnitConversion
00003: from decimal import Decimal
00004:
00005: class Command(BaseCommand):
00006:     help = 'Valida si los precios por unidad están proporcionales según las conversion
es'
00007:
00008:     def handle(self, *args, **kwargs):
00009:         self.stdout.write(self.style.NOTICE("■ Iniciando validación de unidades y prec
ios..."))
00010:
00011:         inconsistencias = 0
00012:
00013:         for pu in ProductUnit.objects.select_related('product', 'unit'):
00014:             base_unit = pu.product.unit_default
00015:             if pu.unit == base_unit:
00016:                 continue  # Nada que comparar con unidad base
00017:
00018:             try:
00019:                 conv = UnitConversion.objects.get(from_unit=pu.unit, to_unit=base_unit
)
00020:                 esperado = Decimal(1) / conv.conversion_factor if conv.sign == '/' els
e Decimal(conv.conversion_factor)
00021:                 if abs(pu.conversion_factor - esperado) > Decimal('0.01'):
00022:                     inconsistencias += 1
00023:                     self.stdout.write(self.style.WARNING(
00024:                         f"■■ {pu.product.name} - {pu.unit.code}: "
00025:                         f"esperado {esperado}, tiene {pu.conversion_factor}"
00026:                     ))
00027:             except UnitConversion.DoesNotExist:
00028:                 self.stdout.write(self.style.WARNING(
00029:                     f"■ No se encontró conversión entre {pu.unit.code} → {base_unit.c
ode} "
00030:                     f"para {pu.product.name}"
00031:                 ))
00032:
00033:         if inconsistencias == 0:
00034:             self.stdout.write(self.style.SUCCESS("■ Todo en orden. No se encontraron i
nconsistencias."))
00035:         else:
00036:             self.stdout.write(self.style.ERROR(f"■ Se encontraron {inconsistencias} po
sibles errores."))
00037:
```

appinventory\migrations\0001_initial.py

```
00001: # Generated by Django 5.0.3 on 2025-04-10 03:12
00002:
00003: import django.db.models.deletion
00004: from django.conf import settings
00005: from django.db import migrations, models
00006:
00007:
00008: class Migration(migrations.Migration):
00009:
00010:     initial = True
00011:
00012:     dependencies = [
00013:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00014:     ]
00015:
00016:     operations = [
00017:         migrations.CreateModel(
00018:             name='PriceType',
00019:             fields=[
00020:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00021:                 ('name', models.CharField(max_length=15, unique=True)),
00022:                 ('description', models.TextField(blank=True)),
00023:                 ('is_active', models.BooleanField(default=True)),
00024:             ],
00025:         ),
00026:         migrations.CreateModel(
00027:             name='ProductBrand',
00028:             fields=[
00029:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00030:                 ('name', models.CharField(max_length=100)),
00031:                 ('is_active', models.BooleanField(default=True)),
00032:             ],
00033:         ),
00034:         migrations.CreateModel(
00035:             name='ProductCategory',
00036:             fields=[
00037:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00038:                 ('name', models.CharField(max_length=100)),
00039:                 ('description', models.TextField(blank=True)),
00040:                 ('is_active', models.BooleanField(default=True)),
00041:             ],
00042:         ),
00043:         migrations.CreateModel(
00044:             name='UnitCategory',
00045:             fields=[
00046:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00047:                 ('name', models.CharField(max_length=50, unique=True)),
00048:                 ('description', models.TextField(blank=True)),
00049:                 ('is_active', models.BooleanField(default=True)),
00050:             ],
00051:         ),
00052:         migrations.CreateModel(
00053:             name='Warehouse',
00054:             fields=[
00055:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
```

appinventory\migrations\0001_initial.py

```
00056:                 ('name', models.CharField(max_length=100)),
00057:                 ('location', models.TextField(blank=True)),
00058:                 ('is_active', models.BooleanField(default=True)),
00059:             ],
00060:         ),
00061:         migrations.CreateModel(
00062:             name='UnitOfMeasure',
00063:             fields=[
00064:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00065:                 ('name', models.CharField(max_length=50, unique=True)),
00066:                 ('code', models.CharField(max_length=10, unique=True)),
00067:                 ('reference_unit', models.BooleanField(default=False, help_text='Unida
d base para conversiones')),
00068:                 ('conversion_sign', models.CharField(choices=[('*', 'Menor que la unid
ad de referencia'), ('/', 'Mayor que la unidad de referencia')], default='*', max_length=1)),
00069:                 ('conversion_factor', models.DecimalField(decimal_places=4, default=1,
 max_digits=10)),
00070:                 ('is_active', models.BooleanField(default=True)),
00071:                 ('category', models.ForeignKey(on_delete=django.db.models.deletion.CAS
CADE, to='appinventory.unitcategory')),
00072:             ],
00073:         ),
00074:         migrations.CreateModel(
00075:             name='Product',
00076:             fields=[
00077:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00078:                 ('name', models.CharField(max_length=255)),
00079:                 ('sku', models.CharField(max_length=100, unique=True)),
00080:                 ('reorder_level', models.DecimalField(decimal_places=2, default=0, max
_digits=10)),
00081:                 ('created_at', models.DateTimeField(auto_now_add=True)),
00082:                 ('is_active', models.BooleanField(default=True)),
00083:                 ('brand', models.ForeignKey(blank=True, null=True, on_delete=django.db
.models.deletion.SET_NULL, to='appinventory.productbrand')),
00084:                 ('category', models.ForeignKey(null=True, on_delete=django.db.models.d
eletion.SET_NULL, to='appinventory.productcategory')),
00085:                 ('unit_default', models.ForeignKey(null=True, on_delete=django.db.mode
ls.deletion.SET_NULL, to='appinventory.unitofmeasure')),
00086:             ],
00087:         ),
00088:         migrations.CreateModel(
00089:             name='InventoryMovement',
00090:             fields=[
00091:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00092:                 ('quantity', models.DecimalField(decimal_places=2, max_digits=12)),
00093:                 ('movement_type', models.SmallIntegerField(choices=[(1, 'Entrada'), (-
1, 'Salida'), (0, 'Ajuste')])),
00094:                 ('reason', models.CharField(blank=True, max_length=255, null=True)),
00095:                 ('document', models.CharField(blank=True, max_length=100, null=True)),
00096:                 ('line_id', models.PositiveIntegerField(blank=True, null=True)),
00097:                 ('timestamp', models.DateTimeField(auto_now_add=True)),
00098:                 ('created_by', models.ForeignKey(blank=True, null=True, on_delete=djan
go.db.models.deletion.SET_NULL, to=settings.AUTH_USER_MODEL)),
00099:                 ('product', models.ForeignKey(on_delete=django.db.models.deletion.CASC
ADE, to='appinventory.product')),
00100:                 ('unit', models.ForeignKey(blank=True, null=True, on_delete=django.db.
models.deletion.SET_NULL, to='appinventory.unitofmeasure')),
```

appinventory\migrations\0001_initial.py

```
00101:                 ('warehouse', models.ForeignKey(on_delete=django.db.models.deletion.CA
SCADE, to='appinventory.warehouse')),
00102:             ],
00103:             options={
00104:                 'ordering': ['-timestamp'],
00105:             },
00106:         ),
00107:         migrations.CreateModel(
00108:             name='ProductUnit',
00109:             fields=[
00110:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00111:                 ('is_purchase', models.BooleanField(default=False)),
00112:                 ('is_sale', models.BooleanField(default=False)),
00113:                 ('product', models.ForeignKey(on_delete=django.db.models.deletion.CASC
ADE, to='appinventory.product')),
00114:                 ('unit', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE
, to='appinventory.unitofmeasure')),
00115:             ],
00116:             options={
00117:                 'unique_together': {('product', 'unit')},
00118:             },
00119:         ),
00120:         migrations.CreateModel(
00121:             name='ProductPrice',
00122:             fields=[
00123:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00124:                 ('price', models.DecimalField(decimal_places=2, max_digits=10)),
00125:                 ('is_default', models.BooleanField(default=False)),
00126:                 ('valid_from', models.DateField(blank=True, null=True)),
00127:                 ('valid_until', models.DateField(blank=True, null=True)),
00128:                 ('is_active', models.BooleanField(default=True)),
00129:                 ('price_type', models.ForeignKey(on_delete=django.db.models.deletion.C
ASCADE, to='appinventory.pricetype')),
00130:                 ('product', models.ForeignKey(on_delete=django.db.models.deletion.CASC
ADE, related_name='prices', to='appinventory.product')),
00131:                 ('unit', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE
, to='appinventory.unitofmeasure')),
00132:             ],
00133:             options={
00134:                 'unique_together': {('product', 'price_type')},
00135:             },
00136:         ),
00137:         migrations.CreateModel(
00138:             name='Stock',
00139:             fields=[
00140:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00141:                 ('quantity', models.DecimalField(decimal_places=2, max_digits=10)),
00142:                 ('product', models.ForeignKey(on_delete=django.db.models.deletion.CASC
ADE, to='appinventory.product')),
00143:                 ('warehouse', models.ForeignKey(on_delete=django.db.models.deletion.CA
SCADE, to='appinventory.warehouse')),
00144:             ],
00145:             options={
00146:                 'unique_together': {('product', 'warehouse')},
00147:             },
00148:         ),
00149:     ]
```

appinventory\migrations\0002_alter_unitofmeasure_conversion_sign.py

```
00001: # Generated by Django 5.0.3 on 2025-04-10 03:48
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('appinventory', '0001_initial'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterField(
00014:             model_name='unitofmeasure',
00015:             name='conversion_sign',
00016:             field=models.CharField(choices=[('ref', 'Reference Unit'), ('*', 'Smaller
than reference'), ('/', 'Greater than reference')], default='ref', max_length=4),
00017:         ),
00018:     ]
```

appinventory\migrations\0003_alter_unitofmeasure_conversion_sign.py

```
00001: # Generated by Django 5.0.3 on 2025-04-21 21:52
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('appinventory', '0002_alter_unitofmeasure_conversion_sign'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterField(
00014:             model_name='unitofmeasure',
00015:             name='conversion_sign',
00016:             field=models.CharField(choices=[('ref', 'Reference Unit'), ('*', 'Smaller
than reference (*)'), ('/', 'Greater than reference (/)')], default='ref', max_length=4),
00017:         ),
00018:     ]
```

appinventory\migrations\0004_alter_productprice_unique_together.py

```
00001: # Generated by Django 5.0.3 on 2025-07-13 12:23
00002:
00003: from django.db import migrations
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('appinventory', '0003_alter_unitofmeasure_conversion_sign'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterUniqueTogether(
00014:             name='productprice',
00015:             unique_together={('product', 'price_type', 'unit', 'valid_from', 'valid_un
til')},
00016:         ),
00017:     ]
```

appinventory\migrations\0005_alter_productunit_unique_together.py

```
00001: # Generated by Django 5.0.3 on 2025-07-13 16:16
00002:
00003: from django.db import migrations
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('appinventory', '0004_alter_productprice_unique_together'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterUniqueTogether(
00014:             name='productunit',
00015:             unique_together=set(),
00016:         ),
00017:     ]
```

appinventory\migrations\0006_productprice_is_sale_delete_productunit.py

```
00001: # Generated by Django 5.0.3 on 2025-07-14 23:47
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('appinventory', '0005_alter_productunit_unique_together'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AddField(
00014:             model_name='productprice',
00015:             name='is_sale',
00016:             field=models.BooleanField(default=False),
00017:         ),
00018:         migrations.DeleteModel(
00019:             name='ProductUnit',
00020:         ),
00021:     ]
```

appinventory\migrations\0007_productprice_is_purchase.py

```
00001: # Generated by Django 5.0.3 on 2025-07-14 23:54
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('appinventory', '0006_productprice_is_sale_delete_productunit'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AddField(
00014:             model_name='productprice',
00015:             name='is_purchase',
00016:             field=models.BooleanField(default=False),
00017:         ),
00018:     ]
```

appinventory\migrations\0008_alter_productbrand_name_alter_productcategory_name_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-08-20 03:23
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('appinventory', '0007_productprice_is_purchase'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterField(
00014:             model_name='productbrand',
00015:             name='name',
00016:             field=models.CharField(max_length=100, unique=True),
00017:         ),
00018:         migrations.AlterField(
00019:             model_name='productcategory',
00020:             name='name',
00021:             field=models.CharField(max_length=100, unique=True),
00022:         ),
00023:         migrations.AlterField(
00024:             model_name='warehouse',
00025:             name='name',
00026:             field=models.CharField(max_length=100, unique=True),
00027:         ),
00028:     ]
```

appinventory\migrations__init__.py

```
00001:
```

appinventory\models.py

```
00001: from django.db import models
00002: from django.conf import settings
00003: from django.core.exceptions import ValidationError
00004: from appinventory.helpers import convert_to_reference_unit
00005:
00006: # Categorías de Unidades de Medida (Longitud, Peso, Volumen...)
00007: class UnitCategory(models.Model):
00008:     name = models.CharField(max_length=50, unique=True)
00009:     description = models.TextField(blank=True)
00010:     is_active = models.BooleanField(default=True)
00011:
00012:     def __str__(self):
00013:         return self.name
00014:
00015:
00016: class UnitOfMeasure(models.Model):
00017:     name = models.CharField(max_length=50, unique=True)
00018:     code = models.CharField(max_length=10, unique=True)
00019:     category = models.ForeignKey(UnitCategory,  on_delete=models.PROTECT)
00020:     reference_unit = models.BooleanField(default=False, help_text="Unidad base para co
nversiones")
00021:
00022:     SIGN_CHOICES = [
00023:         ('ref', 'Reference Unit'),
00024:         ('*', 'Smaller than reference (*)'),
00025:         ('/', 'Greater than reference (/)'),
00026:     ]
00027:     conversion_sign = models.CharField(max_length=4, choices=SIGN_CHOICES, default='re
f')
00028:     conversion_factor = models.DecimalField(max_digits=10, decimal_places=4, default=1
)
00029:     is_active = models.BooleanField(default=True)
00030:
00031:     def __str__(self):
00032:         return f"{self.code} - {self.name}"
00033:
00034:     def clean(self):
00035:         if self.conversion_sign == 'ref':
00036:             existing_ref = UnitOfMeasure.objects.filter(
00037:                 category=self.category,
00038:                 conversion_sign='ref'
00039:             )
00040:             if self.pk:
00041:                 existing_ref = existing_ref.exclude(pk=self.pk)
00042:             if existing_ref.exists():
00043:                 raise ValidationError(
00044:                     "There is already a reference unit for this category.")
00045:
00046:
00047: class Warehouse(models.Model):
00048:     name = models.CharField(max_length=100, unique=True)
00049:     location = models.TextField(blank=True)
00050:     is_active = models.BooleanField(default=True)
00051:
00052:     def __str__(self):
00053:         return self.name
00054:
00055:
00056: class ProductCategory(models.Model):
00057:     name = models.CharField(max_length=100, unique=True)
```

appinventory\models.py

```
00058:     description = models.TextField(blank=True)
00059:     is_active = models.BooleanField(default=True)
00060:
00061:     def __str__(self):
00062:         return self.name
00063:
00064:
00065: class ProductBrand(models.Model):
00066:     name = models.CharField(max_length=100, unique=True)
00067:     is_active = models.BooleanField(default=True)
00068:
00069:     def __str__(self):
00070:         return self.name
00071:
00072:
00073: class Product(models.Model):
00074:     name = models.CharField(max_length=255)
00075:     sku = models.CharField(max_length=100, unique=True)
00076:     category = models.ForeignKey(ProductCategory,  on_delete=models.PROTECT, null=True
)
00077:     brand = models.ForeignKey(ProductBrand,  on_delete=models.PROTECT, null=True, blan
k=True)
00078:     reorder_level = models.DecimalField(max_digits=10, decimal_places=2, default=0)
00079:     unit_default = models.ForeignKey(UnitOfMeasure, on_delete=models.PROTECT, null=Tru
e)
00080:     created_at = models.DateTimeField(auto_now_add=True)
00081:     is_active = models.BooleanField(default=True)
00082:
00083:     def __str__(self):
00084:         return self.name
00085:
00086: class PriceType(models.Model):
00087:     name = models.CharField(max_length=15, unique=True)
00088:     description = models.TextField(blank=True)
00089:     is_active = models.BooleanField(default=True)
00090:
00091:     def __str__(self):
00092:         return self.name
00093:
00094:
00095: class ProductPrice(models.Model):
00096:     product = models.ForeignKey(Product, related_name="prices", on_delete=models.CASCA
DE)
00097:     price_type = models.ForeignKey(PriceType, on_delete=models.PROTECT)
00098:     unit = models.ForeignKey(UnitOfMeasure, on_delete=models.PROTECT)
00099:     price = models.DecimalField(max_digits=10, decimal_places=2)
00100:     is_default = models.BooleanField(default=False)
00101:     is_sale = models.BooleanField(default=False)
00102:     is_purchase = models.BooleanField(default=False)
00103:     valid_from = models.DateField(null=True, blank=True)
00104:     valid_until = models.DateField(null=True, blank=True)
00105:     is_active = models.BooleanField(default=True)
00106:
00107:     class Meta:
00108:         unique_together = ("product", "price_type", "unit", "valid_from", "valid_until
")
00109:
00110:     def clean(self):
00111:         if not self.is_purchase and not self.is_sale:
00112:             raise ValidationError(
```

appinventory\models.py

```
00113:                 "You must indicate whether this unit is for purchasing, selling, or bo
th.")
00114:
00115:
00116:     def __str__(self):
00117:         return f"{self.product} | {self.price_type} | {self.unit} → ${self.price}"
00118:
00119:
00120: class Stock(models.Model):
00121:     product = models.ForeignKey(Product, on_delete=models.CASCADE)
00122:     warehouse = models.ForeignKey(Warehouse, on_delete=models.CASCADE)
00123:     quantity = models.DecimalField(max_digits=10, decimal_places=2)
00124:
00125:     class Meta:
00126:         unique_together = ("product", "warehouse")
00127:
00128:
00129: class InventoryMovement(models.Model):
00130:     MOVEMENT_TYPE_CHOICES = [
00131:         (1, 'Entrada'),
00132:         (-1, 'Salida'),
00133:         (0, 'Ajuste')
00134:     ]
00135:
00136:     product = models.ForeignKey(Product, on_delete=models.PROTECT)
00137:     warehouse = models.ForeignKey(Warehouse, on_delete=models.PROTECT)
00138:     quantity = models.DecimalField(max_digits=12, decimal_places=2)
00139:     movement_type = models.SmallIntegerField(choices=MOVEMENT_TYPE_CHOICES)
00140:     reason = models.CharField(max_length=255, blank=True, null=True)
00141:     unit = models.ForeignKey(UnitOfMeasure, on_delete=models.PROTECT, null=True, blank
=True)
00142:     document = models.CharField(max_length=100, blank=True, null=True)
00143:     line_id = models.PositiveIntegerField(blank=True, null=True)
00144:     timestamp = models.DateTimeField(auto_now_add=True)
00145:     created_by = models.ForeignKey(
00146:         settings.AUTH_USER_MODEL, on_delete=models.SET_NULL, null=True, blank=True)
00147:
00148:     class Meta:
00149:         ordering = ['-timestamp']
00150:
00151:     def __str__(self):
00152:         return f"{self.get_movement_type_display()} - {self.product} ({self.quantity})
 en {self.warehouse}"
00153:
00154:     def get_converted_quantity(self):
00155:         print("self.quantity: ",self.quantity)
00156:         return convert_to_reference_unit(self.product, self.unit, self.quantity)
00157:
00158:     def save(self, *args, **kwargs):
00159:         if not self.product or not self.warehouse:
00160:             raise ValueError("■ No se puede guardar InventoryMovement sin producto o a
lmacén.")
00161:
00162:         print(f"■ Salvando movimiento: product={self.product}, quantity={self.quantity
}, unit={self.unit}, warehouse={self.warehouse}")
00163:
00164:         # Calcular cantidad convertida
00165:         try:
00166:             converted_qty = self.get_converted_quantity() if self.unit else self.quant
ity
```

appinventory\models.py

```
00167:         except Exception as e:
00168:             print(f"■ Error al convertir cantidad: {e}")
00169:             converted_qty = self.quantity
00170:
00171:         if converted_qty is None:
00172:             raise ValueError("■ Error: cantidad convertida terminó en None")
00173:
00174:         # Guardar la cantidad convertida en el mismo campo
00175:         self.quantity = converted_qty
00176:
00177:         # Guardar el movimiento
00178:         print(f"■ Guardando en DB → quantity={self.quantity} (tipo: {type(self.quanti
ty)})")
00179:         super().save(force_insert=not self.pk, *args, **kwargs)
00180:
00181:         # Ajustar el stock
00182:         stock, _ = Stock.objects.get_or_create(product=self.product, warehouse=self.wa
rehouse)
00183:         old_qty = stock.quantity
00184:         stock.quantity += self.quantity * self.movement_type
00185:         stock.save()
00186:
00187:         print(f"■ Stock actualizado: {old_qty} → {stock.quantity} (producto: {self.pr
oduct}, almacén: {self.warehouse})")
00188:
00189:     def delete(self, *args, **kwargs):
00190:         from .models import Stock
00191:         converted_qty = self.get_converted_quantity()
00192:         stock, _ = Stock.objects.get_or_create(product=self.product, warehouse=self.wa
rehouse)
00193:         stock.quantity -= converted_qty * self.movement_type
00194:         stock.save()
00195:         super().delete(*args, **kwargs)
```

appinventory\serializers.py

```
00001: from rest_framework import serializers
00002: from appinventory.models import (
00003:     Warehouse, ProductCategory, ProductBrand, Product, UnitOfMeasure,
00004:     UnitCategory, PriceType, ProductPrice
00005: )
00006: from django.db import transaction
00007:
00008: # Serializador para almacenes
00009: class WarehouseSerializer(serializers.ModelSerializer):
00010:     class Meta:
00011:         model = Warehouse
00012:         fields = '__all__'
00013:
00014: # Serializador para categorías de productos
00015: class ProductCategorySerializer(serializers.ModelSerializer):
00016:     class Meta:
00017:         model = ProductCategory
00018:         fields = '__all__'
00019:
00020: # Serializador para categorías de unidades
00021: class UnitCategorySerializer(serializers.ModelSerializer):
00022:     class Meta:
00023:         model = UnitCategory
00024:         fields = '__all__'
00025:
00026: # Serializador para marcas de productos
00027: class ProductBrandSerializer(serializers.ModelSerializer):
00028:     class Meta:
00029:         model = ProductBrand
00030:         fields = '__all__'
00031:
00032: # Serializador para precios de productos
00033: class ProductPriceSerializer(serializers.ModelSerializer):
00034:     class Meta:
00035:         model = ProductPrice
00036:         exclude = ['product']
00037:
00038: # Serializador principal para productos, incluye relación con precios y unidades
00039: class ProductSerializer(serializers.ModelSerializer):
00040:     prices = ProductPriceSerializer(many=True, required=False)
00041:
00042:     class Meta:
00043:         model = Product
00044:         fields = '__all__'
00045:
00046:     def validate_prices(self, value):
00047:         seen = set()
00048:         for item in value:
00049:             key = (item.get('price_type'), item.get('unit'), item.get('valid_from'), i
tem.get('valid_until'))
00050:             if key in seen:
00051:                 raise serializers.ValidationError(f"Duplicate price for type/unit/peri
od: {key}")
00052:             seen.add(key)
00053:         return value
00054:
00055:     def create(self, validated_data):
00056:         prices_data = validated_data.pop('prices', [])
00057:
00058:         with transaction.atomic():
```

appinventory\serializers.py

```
00059:             product = super().create(validated_data)
00060:
00061:             for price_data in prices_data:
00062:                 ProductPrice.objects.create(product=product, **price_data)
00063:
00064:         return product
00065:
00066:     def update(self, instance, validated_data):
00067:         prices_data = validated_data.pop('prices', [])
00068:
00069:
00070:         # Actualiza campos del producto
00071:         for attr, value in validated_data.items():
00072:             setattr(instance, attr, value)
00073:         instance.save()
00074:
00075:         # Elimina precios anteriores y guarda nuevos
00076:         instance.prices.all().delete()
00077:         for price_data in prices_data:
00078:             ProductPrice.objects.create(product=instance, **price_data)
00079:
00080:         return instance
00081:
00082: # Serializador para unidades de medida
00083: class UnitOfMeasureSerializer(serializers.ModelSerializer):
00084:     category_name = serializers.CharField(source='category.name', read_only=True)
00085:
00086:     class Meta:
00087:         model = UnitOfMeasure
00088:         fields = [
00089:             'id', 'name', 'code', 'category', 'category_name',
00090:             'reference_unit', 'conversion_sign', 'conversion_factor', 'is_active'
00091:         ]
00092:
00093: # Serializador para tipos de precio
00094: class PriceTypeSerializer(serializers.ModelSerializer):
00095:     class Meta:
00096:         model = PriceType
00097:         fields = ['id', 'name', 'description', 'is_active']
00098:
00099: # Serializador compacto para listados
00100: class ProductListSerializer(serializers.ModelSerializer):
00101:     category_name = serializers.CharField(source='category.name', default='', read_onl
y=True)
00102:     brand_name = serializers.CharField(source='brand.name', default='', read_only=True
)
00103:     unit_name = serializers.CharField(source='unit_default.name', default='', read_onl
y=True)
00104:
00105:     class Meta:
00106:         model = Product
00107:         fields = [
00108:             'id', 'name', 'sku', 'category_name', 'brand_name',
00109:             'reorder_level', 'unit_name', 'is_active'
00110:         ]
00111:
00112: # Serializador para detalle completo de producto (usado en modo edición o vista)
00113: class ProductDetailSerializer(ProductSerializer):
00114:     prices = ProductPriceSerializer(many=True, read_only=True)
00115:
```

appinventory\serializers.py

```
00116:     class Meta:
00117:         model = Product
00118:         fields = '__all__'
00119:         extra_fields = ['prices']
```

appinventory\serializers_schema.py

```
00001: from rest_framework import serializers
00002: from appinventory.models import ProductCategory
00003:
00004: class ProductCategorySchemaSerializer(serializers.ModelSerializer):
00005:     class Meta:
00006:         model = ProductCategory
00007:         fields = '__all__'
```

appinventory\templates\admin\base_site.html

```
00001: {% extends "admin/base_site.html" %}
00002:
00003: {% block branding %}
00004:   <h1 id="site-name">
00005:     <a href="{% url 'admin:index' %}">Chalan Admin</a>
00006:   </h1>
00007:   <div style="margin-top: 10px;">
00008:     <a href="{% url 'inventory-dashboard' %}" class="button default" style="margin-top
:5px;">
00009:         ■ Go to Dashboard
00010:       </a>
00011:   </div>
00012: {% endblock %}
```

appinventory\templates\base.html

```
00001: <!DOCTYPE html>
00002: <html lang="en">
00003:
00004: <head>
00005:   <meta charset="UTF-8">
00006:   <title>{% block title %}Chalan-Pro{% endblock %}</title>
00007:   <meta name="viewport" content="width=device-width, initial-scale=1">
00008:   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
 rel="stylesheet">
00009: </head>
00010:
00011: <body>
00012:
00013:   <nav class="navbar navbar-expand-lg navbar-dark bg-dark mb-4">
00014:     <div class="container-fluid">
00015:       <a class="navbar-brand" href="/">Chalan-Pro</a>
00016:     </div>
00017:   </nav>
00018:
00019:   <main class="container">
00020:     {% block content %}{% endblock %}
00021:   </main>
00022:
00023:
00024:   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.m
in.js"></script>
00025: </body>
00026:
00027: </html>
```

appinventory\templates\inventory\dashboard.html

```
00001: {% extends 'base.html' %}
00002: {% block content %}
00003: <div class="container mt-5">
00004:   <h1 class="mb-4">■ {{ page_title }}</h1>
00005:   <a href="/admin/" class="btn btn-outline-dark mb-3">■ Back to Admin</a>
00006:
00007:   <!-- Totals -->
00008:   <div class="row mb-4">
00009:     <div class="col-md-4">
00010:       <div class="card shadow-sm">
00011:         <div class="card-body">
00012:           <h5 class="card-title">Total Products</h5>
00013:           <p class="card-text display-6">{{ total_products }}</p>
00014:         </div>
00015:       </div>
00016:     </div>
00017:     <div class="col-md-4">
00018:       <div class="card shadow-sm">
00019:         <div class="card-body">
00020:           <h5 class="card-title">Total Warehouses</h5>
00021:           <p class="card-text display-6">{{ total_warehouses }}</p>
00022:         </div>
00023:       </div>
00024:     </div>
00025:     <div class="col-md-4">
00026:       <div class="card shadow-sm">
00027:         <div class="card-body">
00028:           <h5 class="card-title">Stock Units</h5>
00029:           <p class="card-text display-6">{{ total_stock_units }}</p>
00030:         </div>
00031:       </div>
00032:     </div>
00033:   </div>
00034:
00035:   <!-- Low Stock Alerts -->
00036:   <h3>■■ Products Below Reorder Level</h3>
00037:   <table class="table table-bordered table-sm mt-3">
00038:     <thead class="table-warning">
00039:       <tr>
00040:         <th>Name</th>
00041:         <th>SKU</th>
00042:         <th>Stock</th>
00043:         <th>Reorder Level</th>
00044:       </tr>
00045:     </thead>
00046:     <tbody>
00047:       {% for p in low_stock_products %}
00048:       <tr>
00049:         <td>{{ p.name }}</td>
00050:         <td>{{ p.sku }}</td>
00051:         <td>{{ p.total_stock|default:0 }}</td>
00052:         <td>{{ p.reorder_level }}</td>
00053:       </tr>
00054:       {% empty %}
00055:       <tr><td colspan="4">■ All products above reorder level</td></tr>
00056:       {% endfor %}
00057:     </tbody>
00058:   </table>
00059:
00060:   <!-- Top 5 Lowest Stock -->
```

appinventory\templates\inventory\dashboard.html

```
00061:   <h3>■ Top 5 Products With Lowest Stock</h3>
00062:   <table class="table table-bordered table-sm mt-3">
00063:     <thead class="table-light">
00064:       <tr>
00065:         <th>Name</th>
00066:         <th>SKU</th>
00067:         <th>Stock</th>
00068:       </tr>
00069:     </thead>
00070:     <tbody>
00071:       {% for p in lowest_stock_products %}
00072:       <tr>
00073:         <td>{{ p.name }}</td>
00074:         <td>{{ p.sku }}</td>
00075:         <td>{{ p.total_stock|default:0 }}</td>
00076:       </tr>
00077:       {% empty %}
00078:       <tr><td colspan="3">No products in stock.</td></tr>
00079:       {% endfor %}
00080:     </tbody>
00081:   </table>
00082: </div>
00083: {% endblock %}
```

appinventory\tests.py

```
00001: from django.test import TestCase
00002:
00003: # Create your tests here.
```

appinventory\urls.py

```
00001: from django.urls import path, include
00002: from rest_framework.routers import DefaultRouter
00003: from .views import (
00004:     DashboardView, WarehouseViewSet, ProductCategoryViewSet,
00005:     ProductBrandViewSet, ProductViewSet,
00006:     UnitOfMeasureViewSet, ProductListAPIView, UnitOfMeasureListAPIView,
00007:     UnitCategoryListAPIView, UnitCategoryViewSet, PriceTypeViewSet,
00008:     ProductDataTableAPIView
00009: )
00010:
00011: from .views_validation import validate_units_of_measure
00012:
00013: from .views_schema import (
00014:     ProductCategorySchemaView, ProductBrandSchemaView,
00015:     UnitOfMeasureSchemaView, UnitCategorySchemaView,
00016:     PriceTypeSchemaView
00017: )
00018:
00019: # Router para ViewSets
00020: router = DefaultRouter()
00021: router.register(r'warehouses', WarehouseViewSet, basename='warehouse')
00022: router.register(r'productcategory', ProductCategoryViewSet)
00023: router.register(r'productbrand', ProductBrandViewSet)
00024: router.register(r'products', ProductViewSet, basename='product')
00025: router.register(r'unitsofmeasure', UnitOfMeasureViewSet, basename='unitofmeasure')
00026: router.register(r'unitcategory', UnitCategoryViewSet)
00027: router.register(r'pricetypes', PriceTypeViewSet)
00028:
00029: urlpatterns = [
00030:     path('dashboard/', DashboardView.as_view(), name='inventory-dashboard'),
00031:     path('api/validate-units-measure/', validate_units_of_measure, name='validate-unit
s-measure'),
00032:
00033:     # Schema endpoints
00034:     path('api/schema/product-category/', ProductCategorySchemaView.as_view(), name='pr
oduct-category-schema'),
00035:     path('api/schema/productcategory/', ProductCategorySchemaView.as_view()),  # alias
 alternativo
00036:     path('api/schema/productbrand/', ProductBrandSchemaView.as_view(), name='productbr
and-schema'),
00037:     path('api/schema/unitofmeasure/', UnitOfMeasureSchemaView.as_view(), name='unitofm
easure-schema'),
00038:     path('api/unitcategories/', UnitCategoryListAPIView.as_view(), name='unitcategory-
list'),
00039:     path('api/schema/unitcategory/', UnitCategorySchemaView.as_view(), name='unitcateg
ory-schema'),
00040:     path('api/schema/pricetype/', PriceTypeSchemaView.as_view(), name='pricetype-schem
a'),
00041:
00042:     # List endpoints
00043:     path('api/products/options/', ProductListAPIView.as_view(), name='product-list-opt
ions'),
00044:     path('api/unitsofmeasure-options/', UnitOfMeasureListAPIView.as_view(), name='unit
ofmeasure-options'),
00045:     path('api/datatable-products/', ProductDataTableAPIView.as_view(), name='datatable
-products'),
00046:
00047:     # CRUD API routes
00048:     path('api/', include(router.urls)),
00049: ]
```

appinventory\views.py

```
00001: # Django core
00002: from django.views.generic import TemplateView
00003: from django.db.models import F, Sum, OuterRef, Subquery
00004: from django.db.models.deletion import ProtectedError
00005: from django.db import IntegrityError
00006: # Django REST Framework (DRF)
00007: from rest_framework.exceptions import ValidationError
00008: from rest_framework import status, viewsets
00009: from rest_framework.generics import ListAPIView
00010: from rest_framework.views import APIView
00011: from rest_framework.response import Response
00012: from rest_framework.authentication import TokenAuthentication
00013: from rest_framework.decorators import permission_classes, action
00014: from rest_framework.permissions import (
00015:     IsAuthenticated,
00016:     DjangoModelPermissions,
00017:     AllowAny,
00018:     IsAuthenticatedOrReadOnly
00019:     )
00020: from utils.datatable import handle_datatable_query
00021: # App Models
00022: from appinventory.models import (
00023:     Product, Stock, Warehouse, ProductCategory,
00024:     ProductBrand, UnitOfMeasure, UnitCategory, PriceType
00025:     )
00026: from apptransactions.models import Document, DocumentLine
00027: # Serializers
00028: from appinventory.serializers import (
00029:     WarehouseSerializer, ProductCategorySerializer, ProductBrandSerializer,
00030:     ProductSerializer, UnitOfMeasureSerializer, UnitCategorySerializer,
00031:     PriceTypeSerializer, ProductListSerializer,ProductDetailSerializer
00032:     )
00033:
00034:
00035: class DashboardView(TemplateView):
00036:     template_name = "inventory/dashboard.html"
00037:
00038:     def get_context_data(self, **kwargs):
00039:         context = super().get_context_data(**kwargs)
00040:
00041:         # Productos con stock bajo el reorder_level
00042:         low_stock_products = (
00043:             Product.objects.filter(is_active=True)
00044:             .annotate(
00045:                 total_stock=Subquery(
00046:                     Stock.objects.filter(product=OuterRef('pk'))
00047:                     .values('product')
00048:                     .annotate(qty=Sum('quantity'))
00049:                     .values('qty')[:1]
00050:                 )
00051:             )
00052:             .filter(total_stock__lt=F('reorder_level'))
00053:         )
00054:
00055:         # Top 5 productos más bajos en stock
00056:         lowest_stock_products = (
00057:             Product.objects.filter(is_active=True)
00058:             .annotate(
00059:                 total_stock=Subquery(
00060:                     Stock.objects.filter(product=OuterRef('pk'))
```

appinventory\views.py

```
00061:                     .values('product')
00062:                     .annotate(qty=Sum('quantity'))
00063:                     .values('qty')[:1]
00064:                 )
00065:             )
00066:             .order_by('total_stock')[:5]
00067:         )
00068:
00069:         # Últimos movimientos (últimos 10 documentos con líneas)
00070:         recent_documents = (
00071:             Document.objects.filter(is_active=True)
00072:             .select_related('document_type', 'warehouse', 'party')
00073:             .prefetch_related('lines')
00074:             .order_by('-date')[:10]
00075:         )
00076:
00077:         # Totales generales
00078:         total_warehouses = Warehouse.objects.filter(is_active=True).count()
00079:         total_products = Product.objects.filter(is_active=True).count()
00080:         total_stock_units = Stock.objects.aggregate(total=Sum('quantity'))['total'] or
 0
00081:
00082:         context.update({
00083:             'low_stock_products': low_stock_products,
00084:             'lowest_stock_products': lowest_stock_products,
00085:             'recent_documents': recent_documents,
00086:             'total_warehouses': total_warehouses,
00087:             'total_products': total_products,
00088:             'total_stock_units': total_stock_units,
00089:             'page_title': "Inventory Dashboard"
00090:         })
00091:
00092:         return context
00093:
00094: class WarehouseViewSet(viewsets.ModelViewSet):
00095:     queryset = Warehouse.objects.all()
00096:     serializer_class = WarehouseSerializer
00097:     authentication_classes = [TokenAuthentication]
00098:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00099:
00100: class ProductCategoryViewSet(viewsets.ModelViewSet):
00101:     queryset = ProductCategory.objects.all()
00102:     serializer_class = ProductCategorySerializer
00103:     authentication_classes = [TokenAuthentication]
00104:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00105:
00106: class UnitCategoryViewSet(viewsets.ModelViewSet):
00107:     queryset = UnitCategory.objects.all()
00108:     serializer_class = UnitCategorySerializer
00109:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00110:     authentication_classes = [TokenAuthentication]
00111:
00112: class UnitCategoryListAPIView(ListAPIView):
00113:     queryset = UnitCategory.objects.all()
00114:     serializer_class = UnitCategorySerializer
00115:     permission_classes = [IsAuthenticated]
00116:     authentication_classes = [TokenAuthentication]
00117:
00118: class ProductBrandViewSet(viewsets.ModelViewSet):
00119:     queryset = ProductBrand.objects.all()
```

appinventory\views.py

```
00120:     serializer_class = ProductBrandSerializer
00121:     authentication_classes = [TokenAuthentication]
00122:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00123:
00124: class PriceTypeViewSet(viewsets.ModelViewSet):
00125:     queryset = PriceType.objects.all()
00126:     serializer_class = PriceTypeSerializer
00127:     permission_classes = [IsAuthenticated]
00128:     authentication_classes = [TokenAuthentication]
00129:
00130: class ProductViewSet(viewsets.ModelViewSet):
00131:     queryset = Product.objects.all()
00132:     authentication_classes = [TokenAuthentication]
00133:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00134:
00135:     def get_serializer_class(self):
00136:         if self.action == 'list':
00137:             return ProductListSerializer  # Para vistas tipo tabla
00138:         elif self.action == 'retrieve':
00139:             return ProductDetailSerializer  # Para vista detalle o edición
00140:         return ProductSerializer  # Para create, update, partial_update
00141:
00142: @permission_classes([AllowAny])
00143: class ProductListAPIView(APIView):
00144:     def get(self, request):
00145:         products = Product.objects.all()
00146:         return Response([
00147:             {"value": p.id, "label": p.name} for p in products
00148:         ])
00149:
00150: class UnitOfMeasureViewSet(viewsets.ModelViewSet):
00151:     queryset = UnitOfMeasure.objects.all()
00152:     serializer_class = UnitOfMeasureSerializer
00153:     authentication_classes = [TokenAuthentication]
00154:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00155:
00156: @permission_classes([AllowAny])
00157: class UnitOfMeasureListAPIView(APIView):
00158:     def get(self, request):
00159:         units = UnitOfMeasure.objects.all()
00160:         return Response([
00161:             {"value": u.id, "label": u.name} for u in units
00162:         ])
00163:
00164: class ProductDataTableAPIView(APIView):
00165:     authentication_classes = [TokenAuthentication]
00166:     permission_classes = [IsAuthenticated]
00167:
00168:     def get(self, request):
00169:         queryset = Product.objects.select_related('category', 'brand', 'unit_default')
00170:         return handle_datatable_query(
00171:             request,
00172:             queryset,
00173:             ProductListSerializer,
00174:             search_fields=['name', 'sku']
00175:         )
```

appinventory\views_schema.py

```
from appinventory.models import UnitOfMeasure
from rest_framework.views import APIView
from rest_framework.response import Response


class ProductCategorySchemaView(APIView):
    def get(self, request):
        schema = {
            "name": {
                "type": "string",
                "label": "Category",
                "required": False
            },
            "description": {
                "type": "textarea",
                "label": "Description",
                "required": False
            },
            "is_active": {
                "type": "boolean",
                "label": "Active",
                "required": False
            }
        }
        return Response(schema)

class ProductBrandSchemaView(APIView):
    def get(self, request):
        schema = {
            "name": {
                "type": "string",
                "label": "Brand Name",
                "required": True
            },
            "is_active": {
                "type": "boolean",
                "label": "Active",
                "required": False
            }
        }
        return Response(schema)

class UnitOfMeasureSchemaView(APIView):
    def get(self, request):
        schema = {
            "name": {
                "type": "string",
                "label": "Name",
                "required": True
            },
            "code": {
                "type": "string",
                "label": "Code",
                "required": True
            },
            "category": {
                "type": "select",
                "label": "Category",
                "required": True,
                "optionsEndpoint": "/api/unitcategories/"
```

appinventory\views_schema.py

```
            },
            "reference_unit": {
                "type": "boolean",
                "label": "Reference Unit",
                "required": False
            },
            "conversion_sign": {
                "type": "select",
                "label": "Conversion Sign",
                "required": True,
                "options": [
                    {"value": choice[0], "label": choice[1]}
                    for choice in UnitOfMeasure.SIGN_CHOICES
                ]
            },
            "conversion_factor": {
                "type": "string",
                "label": "Conversion Factor",
                "required": True
            },
            "is_active": {
                "type": "boolean",
                "label": "Active",
                "required": False
            }
        }
        return Response(schema)

class UnitCategorySchemaView(APIView):
    def get(self, request):
        schema = {
            "name": {
                "type": "string",
                "label": "Category Name",
                "required": True
            },
            "description": {
                "type": "textarea",
                "label": "Description",
                "required": False
            },
            "is_active": {
                "type": "boolean",
                "label": "Active",
                "required": False
            }
        }
        return Response(schema)

class PriceTypeSchemaView(APIView):
    def get(self, request):
        schema = {
            "name": {
                "type": "string",
                "label": "Price Type",
                "required": True
            },
            "description": {
                "type": "textarea",
                "label": "Description",
```

appinventory\views_schema.py

```
00121:                 "required": False
00122:             },
00123:             "is_active": {
00124:                 "type": "boolean",
00125:                 "label": "Active",
00126:                 "required": False
00127:             }
00128:         }
00129:         return Response(schema)
```

appinventory\views_validation.py

```
00001: """
00002: Este módulo contiene la función de validación para las unidades de producto.
00003:
00004: La función validate_units_of_measure revisa todas las relaciones UnitOfMeasure en la b
ase de datos y detecta posibles errores o inconsistencias, como:
00005: - Factores de conversión desproporcionados (mayores a 1000 o menores a 0.01)
00006: - Detecta factores desproporcionados
00007: - Detecta múltiples unidades base en una misma categoría
00008: - Verifica si hay más de una unidad base por categoría
00009:
00010: Devuelve un JsonResponse con una lista de advertencias o errores encontrados, ayudando
 a mantener la integridad de los datos de inventario.
00011: """
00012: from django.http import JsonResponse
00013: from .models import UnitOfMeasure
00014:
00015: def validate_units_of_measure(request):
00016:     errors = []
00017:
00018:     categories = {}
00019:
00020:     for unit in UnitOfMeasure.objects.select_related('category'):
00021:         cat_name = unit.category.name
00022:         unit_code = unit.code
00023:         factor = unit.conversion_factor
00024:         sign = unit.conversion_sign
00025:
00026:         # Detecta factores desproporcionados
00027:         if factor > 1000 or factor < 0.01:
00028:             errors.append({
00029:                 "unit": unit_code,
00030:                 "category": cat_name,
00031:                 "issue": f"■■ El factor de conversión {factor:.4f} parece desproporci
onado."
00032:             })
00033:
00034:         # Detecta múltiples unidades base en una misma categoría
00035:         if sign == 'ref':
00036:             categories.setdefault(cat_name, []).append(unit_code)
00037:
00038:     # Verifica si hay más de una unidad base por categoría
00039:     for cat_name, ref_units in categories.items():
00040:         if len(ref_units) > 1:
00041:             errors.append({
00042:                 "category": cat_name,
00043:                 "issue": f"■ Hay múltiples unidades base (ref) en esta categoría: {',
'.join(ref_units)}"
00044:             })
00045:
00046:     return JsonResponse(errors, safe=False)
```

appschedule__init__.py

```
00001:
```

appschedule\admin.py

```
00001: from django.contrib import admin
00002: from .models import Event, EventDraft, AbsenceReason, EventImage
00003: from django.utils.html import format_html
00004:
00005:
00006: class BaseCrewTitleAdmin(admin.ModelAdmin):
00007:     def crew_title(self, obj):
00008:         if hasattr(obj, 'crew') and obj.crew:
00009:             crew_name = obj.crew.name
00010:             category = obj.crew.category.name if obj.crew.category else "No category"
00011:             return f"{crew_name} ({category})"
00012:         return "No crew assigned"
00013:
00014:     crew_title.short_description = 'Crew'
00015:
00016:
00017: @admin.register(Event)
00018: class EventAdmin(BaseCrewTitleAdmin):
00019:     list_display = ['title', 'date', 'end_dt', 'crew_title', 'builder', 'job', 'house_
model', 'deleted']
00020:     search_fields = ['title']
00021:     list_filter = ['deleted', 'crew__category']
00022:
00023: @admin.register(EventDraft)
00024: class EventDraftAdmin(admin.ModelAdmin):
00025:     list_display = (
00026:         'title', 'date', 'end_dt', 'crew', 'job', 'lot', 'house_model',
00027:         'is_absence', 'extended_service', 'created_by', 'created_at'
00028:     )
00029:     list_filter = ('crew', 'job', 'is_absence', 'extended_service', 'created_at')
00030:     search_fields = ('lot', 'address', 'title', 'description', 'notes')
00031:     date_hierarchy = 'date'
00032:     ordering = ('-date',)
00033:     readonly_fields = ('created_at', 'updated_at')
00034:
00035:     fieldsets = (
00036:         ('Event Info', {
00037:             'fields': (
00038:                 'date', 'end_dt', 'crew', 'job', 'builder', 'house_model',
00039:                 'lot', 'address', 'title', 'description', 'notes'
00040:             )
00041:         }),
00042:         ('Status & Details', {
00043:             'fields': (
00044:                 'extended_service', 'is_absence', 'absence_reason',
00045:             )
00046:         }),
00047:         ('Audit Info', {
00048:             'fields': ('created_by', 'created_at', 'updated_at')
00049:         }),
00050:     )
00051:
00052: @admin.register(AbsenceReason)
00053: class AbsenceReasonAdmin(admin.ModelAdmin):
00054:     list_display = ['name', 'description', 'is_active']
00055:     search_fields = ['name', 'description']
00056:     list_filter = ['is_active']
00057:
00058:
00059: @admin.register(EventImage)
```

appschedule\admin.py

```
00060: class EventImageAdmin(admin.ModelAdmin):
00061:     list_display = ('id','event_id', 'event', 'uploaded_by', 'uploaded_at', 'image_pre
view')
00062:     list_filter = ('uploaded_at', 'uploaded_by')
00063:     search_fields = ('event__title', 'title', 'lot', 'address')
00064:     readonly_fields = ('image_preview', 'uploaded_at')
00065:
00066:     def image_preview(self, obj):
00067:         if obj.image:
00068:             return format_html(
00069:                 '<img src="{}" width="150" style="border-radius:4px; box-shadow: 0 2px
 6px rgba(0,0,0,0.15);"/>',
00070:                 obj.image.url
00071:             )
00072:         return "(No image)"
00073:
00074:     image_preview.short_description = 'Preview'
00075:     autocomplete_fields = ['event']
```

appschedule\apps.py

```
00001: from django.apps import AppConfig
00002:
00003:
00004: class AppscheduleConfig(AppConfig):
00005:     default_auto_field = 'django.db.models.BigAutoField'
00006:     name = 'appschedule'
00007:     verbose_name = 'Schedule Module'
00008:
00009:     def ready(self):
00010:         import appschedule.signals
```

appschedule\consumers.py

```
00001: import json
00002: from channels.generic.websocket import AsyncWebsocketConsumer, WebsocketConsumer
00003: from channels.db import database_sync_to_async
00004: from .models import Event, EventChatMessage
00005: from .serializers import EventChatMessageSerializer
00006:
00007:
00008: class EventConsumer(AsyncWebsocketConsumer):
00009:     async def connect(self):
00010:         self.calendar_group_name = 'calendar_updates'  # Define a group name
00011:         await self.channel_layer.group_add(self.calendar_group_name, self.channel_name
)
00012:         await self.accept()
00013:
00014:     async def disconnect(self, close_code):
00015:         await self.channel_layer.group_discard(self.calendar_group_name, self.channel_
name)
00016:
00017:     async def event_updated(self, event):
00018:         event_data = event['event_data']
00019:
00020:         # Enviar la información del evento actualizado al WebSocket
00021:         await self.send(text_data=json.dumps({
00022:             'type': 'event.updated',
00023:             'event': event_data
00024:         }))
00025:
00026:     async def event_draft_updated(self, event):
00027:         event_data = event['event_data']
00028:
00029:         # Enviar la información del evento actualizado al WebSocket
00030:         await self.send(text_data=json.dumps({
00031:             'type': 'event.updated_draft',
00032:             'event': event_data
00033:         }))
00034:
00035:     async def receive(self, text_data):
00036:         pass
00037:         # text_data_json = json.loads(text_data)
00038:         # message = text_data_json['message']
00039:         #
00040:         # # Enviar el mensaje al grupo
00041:         # await self.channel_layer.group_send(
00042:         #     self.calendar_group_name,
00043:         #     {
00044:         #         'type': 'chat.message',  # Define el tipo de evento
00045:         #         'message': message
00046:         #     }
00047:         # )
00048:
00049:
00050: class EventNoteConsumer(AsyncWebsocketConsumer):
00051:     async def connect(self):
00052:         self.event_id = self.scope['url_route']['kwargs']['pk']
00053:         self.event_group_name = f"event_{self.event_id}_notes"
00054:
00055:         # Join a group specific to the event
00056:         await self.channel_layer.group_add(self.event_group_name, self.channel_name)
00057:         await self.accept()
00058:
```

appschedule\consumers.py

```
00059:     async def disconnect(self, close_code):
00060:         # Leave the event-specific group
00061:         await self.channel_layer.group_discard(self.event_group_name, self.channel_nam
e)
00062:
00063:     async def note_updated(self, event):
00064:         event_data = event['event_data']
00065:         # Enviar la información del evento actualizado al WebSocket
00066:         await self.send(text_data=json.dumps({
00067:             'type': 'note.updated',
00068:             'event': event_data
00069:         }))
00070:
00071:
00072: class EventChatConsumer(AsyncWebsocketConsumer):
00073:     async def connect(self):
00074:         self.event_id = self.scope['url_route']['kwargs']['event_id']
00075:         self.room_group_name = f"schedule_{self.event_id}_chat"
00076:
00077:         await self.channel_layer.group_add(self.room_group_name, self.channel_name)
00078:         await self.accept()
00079:
00080:     async def disconnect(self, close_code):
00081:         await self.channel_layer.group_discard(self.room_group_name, self.channel_name
)
00082:
00083:     # Receive message from WebSocket
00084:     async def receive(self, text_data):
00085:         pass
00086:         # text_data_json = json.loads(text_data)
00087:         # message = text_data_json["message"]
00088:         #
00089:         # # Send message to room group
00090:         # await self.channel_layer.group_send(
00091:         #     self.room_group_name, {"type": "chat.message", "message": message}
00092:         # )
00093:
00094:     async def chat_updated(self, event):
00095:         data = event['data']
00096:         await self.send(text_data=json.dumps({
00097:             'type': 'chat.updated',
00098:             'data': data
00099:         }))
00100:
00101: class UnreadNotificationConsumer(AsyncWebsocketConsumer):
00102:     async def connect(self):
00103:         self.user = self.scope["user"]
00104:         self.user_id = self.scope["url_route"]["kwargs"]["user_id"]
00105:         self.group_name = f"user_{self.user_id}_unread"
00106:
00107:         print(f"[WS-CONNECT] User {self.user.username} joined {self.group_name}")
00108:         await self.channel_layer.group_add(self.group_name, self.channel_name)
00109:         await self.accept()
00110:
00111:     async def disconnect(self, close_code):
00112:         await self.channel_layer.group_discard(self.group_name, self.channel_name)
00113:
00114:     async def unread_updated(self, event):
00115:         await self.send(text_data=json.dumps({
00116:             "type": "unread.updated",
```

appschedule\consumers.py

```
            "event_id": event["event_id"],
            "count": event["count"]
        }))
```

appschedule\filters.py

```
00001: import django_filters
00002: from django.db.models import Q
00003: from .models import EventDraft
00004:
00005:
00006: class EventDraftFilter(django_filters.rest_framework.FilterSet):
00007:     title = django_filters.CharFilter(lookup_expr='icontains')
00008:     start_at = django_filters.DateTimeFilter(method='filter_by_date_range')
00009:     end_at = django_filters.DateTimeFilter(method='filter_by_date_range')
00010:
00011:     class Meta:
00012:         model = EventDraft
00013:         fields = ['title', 'start_at', 'end_at']
00014:
00015:     def filter_by_date_range(self, queryset, name, value):
00016:         if name == 'start_at':
00017:             lookup = 'gte'
00018:         elif name == 'end_at':
00019:             lookup = 'lte'
00020:         else:
00021:             return queryset  # No debería llegar aquí
00022:
00023:         q = Q()
00024:         if self.request.query_params.get('start_at'):
00025:             start_at_param = self.request.query_params.get('start_at')
00026:             q &= (Q(date__lte=start_at_param,
00027:                     end_dt__gte=start_at_param) |  # Empieza antes o en el inicio y te
rmina después o en el inicio
00028:                   Q(date__gte=start_at_param,
00029:                     date__lte=self.request.query_params.get('end_at', start_at_param))
 |  # Empieza dentro del rango
00030:                   Q(end_dt__gte=start_at_param,
00031:                     end_dt__lte=self.request.query_params.get('end_at', start_at_param
)))  # Termina dentro del rango
00032:
00033:         if self.request.query_params.get('end_at'):
00034:             end_at_param = self.request.query_params.get('end_at')
00035:             q &= (Q(date__lte=end_at_param,
00036:                     end_dt__gte=end_at_param) |  # Empieza antes o en el fin y termina
 después o en el fin
00037:                   Q(date__gte=self.request.query_params.get('start_at', end_at_param),
00038:                     date__lte=end_at_param) |  # Empieza dentro del rango
00039:                   Q(end_dt__gte=self.request.query_params.get('start_at', end_at_param
),
00040:                     end_dt__lte=end_at_param))  # Termina dentro del rango
00041:
00042:         return queryset.filter(q).distinct()
```

appschedule\management\commands\check_duplicates.py

```
00001: from django.core.management.base import BaseCommand
00002: from appschedule.models import Event
00003: from django.db.models import Count, Q
00004:
00005: class Command(BaseCommand):
00006:     help = "Check for duplicate Events by (crew.category, job, lot) or (crew.category,
 job, address)"
00007:
00008:     def handle(self, *args, **kwargs):
00009:         print("\n■ Checking for duplicate Events by crew category + job + lot...")
00010:
00011:         duplicates_lot = (
00012:             Event.objects
00013:             .filter(lot__isnull=False, lot__gt="", deleted=False)
00014:             .values('crew__category', 'job', 'lot')
00015:             .annotate(count=Count('id'))
00016:             .filter(count__gt=1)
00017:         )
00018:
00019:         if duplicates_lot:
00020:             self.stdout.write(self.style.ERROR("\n■ Duplicates found for (crew.categor
y, job, lot):"))
00021:             for dup in duplicates_lot:
00022:                 self.stdout.write(f" - {dup}")
00023:         else:
00024:             self.stdout.write(self.style.SUCCESS("■ No duplicates found for (crew.cate
gory, job, lot)"))
00025:
00026:         print("\n■ Checking for duplicate Events by crew category + job + address...")
00027:
00028:         duplicates_address = (
00029:             Event.objects
00030:             .filter(address__isnull=False, address__gt="", lot__isnull=True, deleted=F
alse)
00031:             .values('crew__category', 'job', 'address')
00032:             .annotate(count=Count('id'))
00033:             .filter(count__gt=1)
00034:         )
00035:
00036:         if duplicates_address:
00037:             self.stdout.write(self.style.ERROR("\n■ Duplicates found for (crew.categor
y, job, address):"))
00038:             for dup in duplicates_address:
00039:                 self.stdout.write(f" - {dup}")
00040:         else:
00041:             self.stdout.write(self.style.SUCCESS("■ No duplicates found for (crew.cate
gory, job, address)"))
00042:
00043:         print("\n■ Done!")
```

appschedule\migrations\0001_initial.py

```
00001: # Generated by Django 5.0.3 on 2025-05-26 05:12
00002:
00003: import django.db.models.deletion
00004: from django.conf import settings
00005: from django.db import migrations, models
00006:
00007:
00008: class Migration(migrations.Migration):
00009:
00010:     initial = True
00011:
00012:     dependencies = [
00013:         ('crewsapp', '0003_category_alter_crew_jobs_alter_crew_members_and_more'),
00014:         ('ctrctsapp', '0012_contract_doc_type_contract_needs_reprint'),
00015:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00016:     ]
00017:
00018:     operations = [
00019:         migrations.CreateModel(
00020:             name='Event',
00021:             fields=[
00022:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00023:                 ('date', models.DateField()),
00024:                 ('end_dt', models.DateField()),
00025:                 ('lot', models.CharField(blank=True, max_length=255, null=True)),
00026:                 ('address', models.CharField(blank=True, max_length=255, null=True)),
00027:                 ('title', models.CharField(blank=True, max_length=255, null=True)),
00028:                 ('description', models.TextField(blank=True, null=True)),
00029:                 ('notes', models.TextField(blank=True, null=True)),
00030:                 ('extended_service', models.BooleanField(default=False)),
00031:                 ('created_at', models.DateTimeField(auto_now_add=True)),
00032:                 ('updated_at', models.DateTimeField(auto_now=True)),
00033:                 ('deleted', models.BooleanField(default=False)),
00034:                 ('builder', models.ForeignKey(blank=True, null=True, on_delete=django.
db.models.deletion.CASCADE, related_query_name='events', to='ctrctsapp.builder')),
00035:                 ('created_by', models.ForeignKey(null=True, on_delete=django.db.models
.deletion.SET_NULL, to=settings.AUTH_USER_MODEL)),
00036:                 ('crew', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE
, to='crewsapp.crew', verbose_name='Crew')),
00037:                 ('house_model', models.ForeignKey(blank=True, null=True, on_delete=dja
ngo.db.models.deletion.SET_NULL, related_query_name='events', to='ctrctsapp.housemodel')),
00038:                 ('job', models.ForeignKey(blank=True, null=True, on_delete=django.db.m
odels.deletion.SET_NULL, related_query_name='events', to='ctrctsapp.job')),
00039:             ],
00040:             options={
00041:                 'verbose_name': 'Event',
00042:                 'verbose_name_plural': 'Events',
00043:             },
00044:         ),
00045:         migrations.CreateModel(
00046:             name='EventChatMessage',
00047:             fields=[
00048:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00049:                 ('message', models.TextField()),
00050:                 ('timestamp', models.DateTimeField(auto_now_add=True)),
00051:                 ('event', models.ForeignKey(on_delete=django.db.models.deletion.CASCAD
E, related_name='chat_messages', to='appschedule.event')),
00052:                 ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE
```

appschedule\migrations\0001_initial.py

```
, to=settings.AUTH_USER_MODEL)),
00053:             ],
00054:         ),
00055:         migrations.CreateModel(
00056:             name='EventDraft',
00057:             fields=[
00058:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00059:                 ('date', models.DateField()),
00060:                 ('end_dt', models.DateField()),
00061:                 ('lot', models.CharField(blank=True, max_length=255, null=True)),
00062:                 ('address', models.CharField(blank=True, max_length=255, null=True)),
00063:                 ('title', models.CharField(blank=True, max_length=255, null=True)),
00064:                 ('description', models.TextField(blank=True, null=True)),
00065:                 ('notes', models.TextField(blank=True, null=True)),
00066:                 ('extended_service', models.BooleanField(default=False)),
00067:                 ('created_at', models.DateTimeField(auto_now_add=True)),
00068:                 ('updated_at', models.DateTimeField(auto_now=True)),
00069:                 ('builder', models.ForeignKey(blank=True, null=True, on_delete=django.
db.models.deletion.CASCADE, related_query_name='drafts', to='ctrctsapp.builder')),
00070:                 ('created_by', models.ForeignKey(null=True, on_delete=django.db.models
.deletion.SET_NULL, to=settings.AUTH_USER_MODEL)),
00071:                 ('crew', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE
, related_query_name='drafts', to='crewsapp.crew')),
00072:                 ('event', models.OneToOneField(blank=True, null=True, on_delete=django
.db.models.deletion.CASCADE, to='appschedule.event', verbose_name='events_draft')),
00073:                 ('house_model', models.ForeignKey(blank=True, null=True, on_delete=dja
ngo.db.models.deletion.CASCADE, related_query_name='drafts', to='ctrctsapp.housemodel')),
00074:                 ('job', models.ForeignKey(blank=True, null=True, on_delete=django.db.m
odels.deletion.CASCADE, related_query_name='drafts', to='ctrctsapp.job')),
00075:             ],
00076:             options={
00077:                 'verbose_name': 'Event Draft',
00078:                 'verbose_name_plural': 'Events Draft',
00079:             },
00080:         ),
00081:         migrations.CreateModel(
00082:             name='EventNote',
00083:             fields=[
00084:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00085:                 ('notes', models.TextField(blank=True, null=True)),
00086:                 ('created_at', models.DateTimeField(auto_now_add=True)),
00087:                 ('updated_at', models.DateTimeField(auto_now=True)),
00088:                 ('event', models.OneToOneField(on_delete=django.db.models.deletion.CAS
CADE, to='appschedule.event', verbose_name='event_note')),
00089:                 ('updated_by', models.ForeignKey(null=True, on_delete=django.db.models
.deletion.SET_NULL, to=settings.AUTH_USER_MODEL)),
00090:             ],
00091:         ),
00092:     ]
```

appschedule\migrations\0002_absencereason_event_absence_reason.py

```
00001: # Generated by Django 5.0.3 on 2025-05-29 00:26
00002:
00003: import django.db.models.deletion
00004: from django.db import migrations, models
00005:
00006:
00007: class Migration(migrations.Migration):
00008:
00009:     dependencies = [
00010:         ('appschedule', '0001_initial'),
00011:     ]
00012:
00013:     operations = [
00014:         migrations.CreateModel(
00015:             name='AbsenceReason',
00016:             fields=[
00017:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00018:                 ('name', models.CharField(max_length=100, unique=True)),
00019:                 ('description', models.TextField(blank=True, null=True)),
00020:                 ('is_active', models.BooleanField(default=True)),
00021:             ],
00022:         ),
00023:         migrations.AddField(
00024:             model_name='event',
00025:             name='absence_reason',
00026:             field=models.ForeignKey(blank=True, null=True, on_delete=django.db.models.
deletion.SET_NULL, to='appschedule.absencereason'),
00027:         ),
00028:     ]
```

appschedule\migrations\0003_alter_absencereason_options_event_is_absence_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-05-31 02:50
00002:
00003: import django.db.models.deletion
00004: from django.db import migrations, models
00005:
00006:
00007: class Migration(migrations.Migration):
00008:
00009:     dependencies = [
00010:         ('appschedule', '0002_absencereason_event_absence_reason'),
00011:     ]
00012:
00013:     operations = [
00014:         migrations.AlterModelOptions(
00015:             name='absencereason',
00016:             options={'verbose_name': 'Absence Reasons', 'verbose_name_plural': 'Absenc
e Reasons'},
00017:         ),
00018:         migrations.AddField(
00019:             model_name='event',
00020:             name='is_absence',
00021:             field=models.BooleanField(default=False),
00022:         ),
00023:         migrations.AddField(
00024:             model_name='eventdraft',
00025:             name='absence_reason',
00026:             field=models.ForeignKey(blank=True, null=True, on_delete=django.db.models.
deletion.SET_NULL, to='appschedule.absencereason'),
00027:         ),
00028:         migrations.AddField(
00029:             model_name='eventdraft',
00030:             name='is_absence',
00031:             field=models.BooleanField(default=False),
00032:         ),
00033:     ]
```

appschedule\migrations\0004_eventchatreadstatus.py

```
00001: # Generated by Django 5.0.3 on 2025-06-19 01:40
00002:
00003: import django.db.models.deletion
00004: from django.conf import settings
00005: from django.db import migrations, models
00006:
00007:
00008: class Migration(migrations.Migration):
00009:
00010:     dependencies = [
00011:         ('appschedule', '0003_alter_absencereason_options_event_is_absence_and_more'),
00012:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00013:     ]
00014:
00015:     operations = [
00016:         migrations.CreateModel(
00017:             name='EventChatReadStatus',
00018:             fields=[
00019:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00020:                 ('last_read', models.DateTimeField(auto_now=True)),
00021:                 ('event', models.ForeignKey(on_delete=django.db.models.deletion.CASCAD
E, related_name='read_statuses', to='appschedule.event')),
00022:                 ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE
, to=settings.AUTH_USER_MODEL)),
00023:             ],
00024:             options={
00025:                 'verbose_name': 'Event Chat Read Status',
00026:                 'verbose_name_plural': 'Event Chat Read Statuses',
00027:                 'unique_together': {('event', 'user')},
00028:             },
00029:         ),
00030:     ]
```

appschedule\migrations\0005_rename_user_eventchatmessage_author_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-06-25 04:06
00002:
00003: import django.utils.timezone
00004: from django.db import migrations, models
00005:
00006:
00007: class Migration(migrations.Migration):
00008:
00009:     dependencies = [
00010:         ('appschedule', '0004_eventchatreadstatus'),
00011:     ]
00012:
00013:     operations = [
00014:         migrations.RenameField(
00015:             model_name='eventchatmessage',
00016:             old_name='user',
00017:             new_name='author',
00018:         ),
00019:         migrations.AlterField(
00020:             model_name='eventchatreadstatus',
00021:             name='last_read',
00022:             field=models.DateTimeField(default=django.utils.timezone.now),
00023:         ),
00024:     ]
```

appschedule\migrations\0006_alter_eventchatreadstatus_unique_together_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-06-26 02:34
00002:
00003: import django.db.models.deletion
00004: from django.conf import settings
00005: from django.db import migrations, models
00006:
00007:
00008: class Migration(migrations.Migration):
00009:
00010:     dependencies = [
00011:         ('appschedule', '0005_rename_user_eventchatmessage_author_and_more'),
00012:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00013:     ]
00014:
00015:     operations = [
00016:         migrations.AlterUniqueTogether(
00017:             name='eventchatreadstatus',
00018:             unique_together={('user', 'message')},
00019:         ),
00020:         migrations.AddField(
00021:             model_name='eventchatreadstatus',
00022:             name='message',
00023:             field=models.ForeignKey(default='2024-07-26 13:43:20.069861', on_delete=dj
ango.db.models.deletion.CASCADE, related_name='read_statuses', to='appschedule.eventchatmessa
ge'),
00024:             preserve_default=False,
00025:         ),
00026:         migrations.AddField(
00027:             model_name='eventchatreadstatus',
00028:             name='read_at',
00029:             field=models.DateTimeField(auto_now=True),
00030:         ),
00031:         migrations.RemoveField(
00032:             model_name='eventchatreadstatus',
00033:             name='event',
00034:         ),
00035:         migrations.RemoveField(
00036:             model_name='eventchatreadstatus',
00037:             name='last_read',
00038:         ),
00039:     ]
```

appschedule\migrations\0007_eventimage.py

```
00001: # Generated by Django 5.0.3 on 2025-07-18 23:33
00002:
00003: import appschedule.models
00004: import django.db.models.deletion
00005: from django.conf import settings
00006: from django.db import migrations, models
00007:
00008:
00009: class Migration(migrations.Migration):
00010:
00011:     dependencies = [
00012:         ('appschedule', '0006_alter_eventchatreadstatus_unique_together_and_more'),
00013:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00014:     ]
00015:
00016:     operations = [
00017:         migrations.CreateModel(
00018:             name='EventImage',
00019:             fields=[
00020:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00021:                 ('image', models.ImageField(upload_to=appschedule.models.EventImageUpl
oadTo())),
00022:                 ('uploaded_at', models.DateTimeField(auto_now_add=True)),
00023:                 ('event', models.ForeignKey(on_delete=django.db.models.deletion.CASCAD
E, related_name='images', to='appschedule.event')),
00024:                 ('uploaded_by', models.ForeignKey(blank=True, null=True, on_delete=dja
ngo.db.models.deletion.SET_NULL, to=settings.AUTH_USER_MODEL)),
00025:             ],
00026:         ),
00027:     ]
```

appschedule\migrations\0008_event_unique_event_crew_job_lot_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-07-19 03:10
00002:
00003: from django.conf import settings
00004: from django.db import migrations, models
00005:
00006:
00007: class Migration(migrations.Migration):
00008:
00009:     dependencies = [
00010:         ('appschedule', '0007_eventimage'),
00011:         ('crewsapp', '0003_category_alter_crew_jobs_alter_crew_members_and_more'),
00012:         ('ctrctsapp', '0012_contract_doc_type_contract_needs_reprint'),
00013:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00014:     ]
00015:
00016:     operations = [
00017:         migrations.AddConstraint(
00018:             model_name='event',
00019:             constraint=models.UniqueConstraint(condition=models.Q(('lot__isnull', Fals
e)), fields=('crew', 'job', 'lot'), name='unique_event_crew_job_lot'),
00020:         ),
00021:         migrations.AddConstraint(
00022:             model_name='event',
00023:             constraint=models.UniqueConstraint(condition=models.Q(('address__isnull',
False), ('lot__isnull', True)), fields=('crew', 'job', 'address'), name='unique_event_crew_jo
b_address'),
00024:         ),
00025:         migrations.AddConstraint(
00026:             model_name='eventdraft',
00027:             constraint=models.UniqueConstraint(condition=models.Q(('lot__isnull', Fals
e)), fields=('crew', 'job', 'lot'), name='unique_eventdraft_crew_job_lot'),
00028:         ),
00029:         migrations.AddConstraint(
00030:             model_name='eventdraft',
00031:             constraint=models.UniqueConstraint(condition=models.Q(('address__isnull',
False), ('lot__isnull', True)), fields=('crew', 'job', 'address'), name='unique_eventdraft_cr
ew_job_address'),
00032:         ),
00033:     ]
```

appschedule\migrations\0009_remove_event_unique_event_crew_job_lot_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-08-19 23:19
00002:
00003: import django.db.models.deletion
00004: from django.db import migrations, models
00005:
00006:
00007: class Migration(migrations.Migration):
00008:
00009:     dependencies = [
00010:         ('appschedule', '0008_event_unique_event_crew_job_lot_and_more'),
00011:     ]
00012:
00013:     operations = [
00014:         migrations.RemoveConstraint(
00015:             model_name='event',
00016:             name='unique_event_crew_job_lot',
00017:         ),
00018:         migrations.RemoveConstraint(
00019:             model_name='event',
00020:             name='unique_event_crew_job_address',
00021:         ),
00022:         migrations.RemoveConstraint(
00023:             model_name='eventdraft',
00024:             name='unique_eventdraft_crew_job_lot',
00025:         ),
00026:         migrations.RemoveConstraint(
00027:             model_name='eventdraft',
00028:             name='unique_eventdraft_crew_job_address',
00029:         ),
00030:         migrations.AlterField(
00031:             model_name='eventnote',
00032:             name='event',
00033:             field=models.OneToOneField(on_delete=django.db.models.deletion.CASCADE, re
lated_name='note', to='appschedule.event', verbose_name='event_note'),
00034:         ),
00035:     ]
```

appschedule\migrations__init__.py

```
00001:
```

appschedule\models.py

```
00001: from django.db import models
00002: from django.contrib.auth.models import User
00003: from crewsapp.models import Crew
00004: from ctrctsapp.models import Builder, HouseModel, Job
00005: from django.utils import timezone    #OAHP
00006: from django.utils.deconstruct import deconstructible
00007: from django.core.exceptions import ValidationError
00008: import os
00009:
00010:
00011: class AbsenceReason(models.Model):
00012:     name = models.CharField(max_length=100, unique=True)
00013:     description = models.TextField(blank=True, null=True)
00014:     is_active = models.BooleanField(default=True)
00015:
00016:     def __str__(self):
00017:         return self.name
00018:
00019:     class Meta:
00020:         verbose_name = "Absence Reasons"
00021:         verbose_name_plural = "Absence Reasons"
00022:         # ordering = ['-date', 'title']
00023:
00024:
00025: class Event(models.Model):
00026:     """
00027:     A model representing an event with details related to a construction job.
00028:     """
00029:     date = models.DateField()
00030:     end_dt = models.DateField()
00031:     crew = models.ForeignKey(Crew, on_delete=models.CASCADE, verbose_name='Crew')
00032:     builder = models.ForeignKey(Builder, on_delete=models.CASCADE, related_query_name=
'events', blank=True, null=True)
00033:     job = models.ForeignKey(Job, on_delete=models.SET_NULL, related_query_name='events
', blank=True, null=True)
00034:     house_model = models.ForeignKey(HouseModel, on_delete=models.SET_NULL, related_que
ry_name='events', blank=True, null=True)
00035:     lot = models.CharField(max_length=255, blank=True, null=True)
00036:     address = models.CharField(max_length=255, blank=True, null=True)
00037:     title = models.CharField(max_length=255, blank=True, null=True)
00038:     description = models.TextField(blank=True, null=True)
00039:     notes = models.TextField(blank=True, null=True)
00040:     extended_service = models.BooleanField(default=False)
00041:     created_at = models.DateTimeField(auto_now_add=True)
00042:     is_absence = models.BooleanField(default=False)
00043:     absence_reason = models.ForeignKey(AbsenceReason, null=True, blank=True, on_delete
=models.SET_NULL)
00044:     updated_at = models.DateTimeField(auto_now=True)
00045:     created_by = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
00046:     deleted = models.BooleanField(default=False)
00047:
00048:     def __str__(self):
00049:         return self.title
00050:
00051:     def clean(self):
00052:         if self.is_absence:
00053:             return
00054:
00055:         if not self.lot and not self.address:
00056:             raise ValidationError("Address or lot/job must be provided")
```

appschedule\models.py

```
00057:
00058:         category = self.crew.category if self.crew else None
00059:         if not category:
00060:             raise ValidationError("Crew category is required")
00061:
00062:         qs_ed = EventDraft.objects.filter(crew__category=category)
00063:         qs_e = Event.objects.filter(crew__category=category, deleted=False)
00064:
00065:         if self.lot:
00066:             qs_ed = qs_ed.filter(job=self.job, lot=self.lot)
00067:             qs_e = qs_e.filter(job=self.job, lot=self.lot)
00068:         elif self.address:
00069:             qs_ed = qs_ed.filter(address=self.address)
00070:             qs_e = qs_e.filter(address=self.address)
00071:
00072:         if self.pk:
00073:             qs_e = qs_e.exclude(pk=self.pk)
00074:         qs_ed = qs_ed.exclude(event_id=self.pk)
00075:
00076:         if qs_ed.exists() or qs_e.exists():
00077:             raise ValidationError("Duplicate Event Detected")
00078:
00079:     def save(self, *args, **kwargs):
00080:         self.title = self.title.upper()
00081:         super().save(*args, **kwargs)
00082:
00083:     class Meta:
00084:         verbose_name = "Event"
00085:         verbose_name_plural = "Events"
00086:         # ordering = ['-date', 'title']
00087:         # Constraint está definido para PostgreSQL ■
00088:         #constraints = [
00089:         #    models.UniqueConstraint(
00090:         #        fields=['crew', 'job', 'lot'],
00091:         #        condition=models.Q(lot__isnull=False),
00092:         #        name='unique_event_crew_job_lot'
00093:         #    ),
00094:         #    models.UniqueConstraint(
00095:         #        fields=['crew', 'job', 'address'],
00096:         #        condition=models.Q(lot__isnull=True, address__isnull=False),
00097:         #        name='unique_event_crew_job_address'
00098:         #    ),
00099:         #]
00100:
00101:
00102: class EventDraft(models.Model):
00103:     """
00104:     A model representing an event draft with details related to a construction job.
00105:     """
00106:     event = models.OneToOneField(Event, on_delete=models.CASCADE, verbose_name='events
_draft', null=True, blank=True)
00107:     date = models.DateField()
00108:     end_dt = models.DateField()
00109:     crew = models.ForeignKey(Crew, on_delete=models.CASCADE, related_query_name='draft
s')
00110:     builder = models.ForeignKey(Builder, on_delete=models.CASCADE, related_query_name=
'drafts', blank=True, null=True)
00111:     job = models.ForeignKey(Job, on_delete=models.CASCADE, related_query_name='drafts'
, blank=True, null=True)
00112:     house_model = models.ForeignKey(HouseModel, on_delete=models.CASCADE, related_quer
```

appschedule\models.py

```
y_name='drafts', blank=True, null=True)
00113:     lot = models.CharField(max_length=255, blank=True, null=True)
00114:     address = models.CharField(max_length=255, blank=True, null=True)
00115:     title = models.CharField(max_length=255, blank=True, null=True)
00116:     description = models.TextField(blank=True, null=True)
00117:     notes = models.TextField(blank=True, null=True)
00118:     extended_service = models.BooleanField(default=False)
00119:     is_absence = models.BooleanField(default=False)
00120:     absence_reason = models.ForeignKey(AbsenceReason, null=True, blank=True, on_delete
=models.SET_NULL)
00121:     created_at = models.DateTimeField(auto_now_add=True)
00122:     updated_at = models.DateTimeField(auto_now=True)
00123:     created_by = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
00124:
00125:     def __str__(self):
00126:         return self.title
00127:
00128:     def clean(self):
00129:         if self.is_absence:
00130:             return
00131:
00132:         if not self.lot and not self.address:
00133:             raise ValidationError("Address or lot/job must be provided")
00134:
00135:         category = self.crew.category if self.crew else None
00136:         if not category:
00137:             raise ValidationError("Crew category is required")
00138:
00139:         qs_ed = EventDraft.objects.filter(crew__category=category)
00140:         qs_e = Event.objects.filter(crew__category=category, deleted=False)
00141:
00142:         if self.lot:
00143:             qs_ed = qs_ed.filter(job=self.job, lot=self.lot)
00144:             qs_e = qs_e.filter(job=self.job, lot=self.lot)
00145:         elif self.address:
00146:             qs_ed = qs_ed.filter(address=self.address)
00147:             qs_e = qs_e.filter(address=self.address)
00148:
00149:         if self.pk:
00150:             qs_ed = qs_ed.exclude(pk=self.pk)
00151:         if self.event_id:
00152:             qs_e = qs_e.exclude(pk=self.event_id)
00153:
00154:         if qs_ed.exists() or qs_e.exists():
00155:             raise ValidationError("Duplicate Event Detected")
00156:
00157:     class Meta:
00158:         verbose_name = "Event Draft"
00159:         verbose_name_plural = "Events Draft"
00160:         # ordering = ['-date', 'title']
00161:         # Constraint está definido para PostgreSQL ■
00162:         #constraints = [
00163:         #    models.UniqueConstraint(
00164:         #        fields=['crew', 'job', 'lot'],
00165:         #        condition=models.Q(lot__isnull=False),
00166:         #        name='unique_eventdraft_crew_job_lot'
00167:         #    ),
00168:         #    models.UniqueConstraint(
00169:         #        fields=['crew', 'job', 'address'],
00170:         #        condition=models.Q(lot__isnull=True, address__isnull=False),
```

appschedule\models.py

```
00171:         #        name='unique_eventdraft_crew_job_address'
00172:         #    ),
00173:         #]
00174:
00175:
00176: class EventNote(models.Model):
00177:     event = models.OneToOneField(Event, on_delete=models.CASCADE, verbose_name='event_
note', related_name='note')
00178:     notes = models.TextField(blank=True, null=True)
00179:     created_at = models.DateTimeField(auto_now_add=True)
00180:     updated_at = models.DateTimeField(auto_now=True)
00181:     updated_by = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
00182:
00183:     def __str__(self):
00184:         return self.notes
00185:
00186:
00187: class EventChatMessage(models.Model):
00188:     event = models.ForeignKey(Event, on_delete=models.CASCADE, related_name='chat_mess
ages')
00189:     author = models.ForeignKey(User, on_delete=models.CASCADE)
00190:     message = models.TextField()
00191:     timestamp = models.DateTimeField(auto_now_add=True)
00192:
00193:     def __str__(self):
00194:         return f"{self.author.username}: {self.message[:50]}"
00195:
00196:
00197: class EventChatReadStatus(models.Model):
00198:     user = models.ForeignKey(User, on_delete=models.CASCADE)
00199:     message = models.ForeignKey('EventChatMessage', on_delete=models.CASCADE, related_
name='read_statuses')
00200:     read_at = models.DateTimeField(auto_now=True)
00201:
00202:     class Meta:
00203:         unique_together = ('user', 'message')
00204:         verbose_name = "Event Chat Read Status"
00205:         verbose_name_plural = "Event Chat Read Statuses"
00206:
00207:     def __str__(self):
00208:         return f"{self.user.username} read message {self.message.id} at {self.read_at}
"
00209:
00210:
00211: @deconstructible
00212: class EventImageUploadTo:
00213:     def __call__(self, instance, filename):
00214:         # Guarda en: event_images/<event_id o 'unassigned'>/filename
00215:         event_id = instance.event.id if instance.event_id else 'unassigned'
00216:         base, ext = os.path.splitext(filename)
00217:         return f'event_images/{event_id}/{base}{ext}'
00218:
00219: class EventImage(models.Model):
00220:     event = models.ForeignKey(Event, on_delete=models.CASCADE, related_name='images')
00221:     image = models.ImageField(upload_to=EventImageUploadTo())
00222:     uploaded_at = models.DateTimeField(auto_now_add=True)
00223:     uploaded_by = models.ForeignKey(User, on_delete=models.SET_NULL, null=True, blank=
True)
00224:
00225:     def __str__(self):
```

appschedule\models.py

```
00226:         return f"Image for Event {self.event_id} ({self.id})"
```

appschedule\routing.py

```
00001: from django.urls import re_path
00002: from . import consumers
00003: from .consumers import UnreadNotificationConsumer
00004:
00005: websocket_urlpatterns = [
00006:     re_path(r"ws/calendar-updates/$", consumers.EventConsumer.as_asgi()),
00007:     re_path(r"ws/schedule/event/(?P<pk>\d+)/$", consumers.EventNoteConsumer.as_asgi())
,
00008:     re_path(r"ws/schedule/event/(?P<event_id>\d+)/chat/$", consumers.EventChatConsumer
.as_asgi()),
00009:     re_path(r'ws/schedule/unread/user/(?P<user_id>\d+)/$', UnreadNotificationConsumer.
as_asgi()),
00010: ]
```

appschedule\serializers.py

```
00001: from datetime import timedelta
00002: from django.db.models import Q
00003: from django.contrib.auth.models import User
00004: from rest_framework import serializers
00005: from rest_framework.validators import ValidationError
00006: from .models import (
00007:     Event, EventDraft, EventNote, EventChatMessage,
00008:     AbsenceReason,EventImage
00009: )
00010:
00011: class EventSerializer(serializers.ModelSerializer):
00012:     """Serializer for events objects"""
00013:     crew_title = serializers.SerializerMethodField()
00014:     crew_category = serializers.SerializerMethodField()
00015:     images = serializers.SerializerMethodField()
00016:
00017:     def get_crew_title(self, obj):
00018:         return obj.crew.name
00019:
00020:     def get_crew_category(self, obj):
00021:         if obj.crew and obj.crew.category:
00022:             return obj.crew.category.name
00023:         return None
00024:
00025:     def get_images(self, obj):
00026:         images = obj.images.all()
00027:         return EventImageSerializer(images, many=True, context=self.context).data
00028:
00029:     class Meta:
00030:         model = Event
00031:         fields = '__all__'
00032:         extra_fields = ['crew_title', 'crew_category', 'images']
00033:
00034:
00035: class EventDraftSerializer(serializers.ModelSerializer):
00036:     """Serializer for events draft objects"""
00037:
00038:     crew_title = serializers.SerializerMethodField()
00039:
00040:     def get_crew_title(self, obj):
00041:         return obj.crew.name
00042:
00043:     class Meta:
00044:         model = EventDraft
00045:         fields = '__all__'
00046:         extra_kwargs = {
00047:             'end_dt': {'required': False, 'allow_null': True},
00048:             'title': {'required': True, 'allow_null': False, 'allow_blank': False},
00049:         }
00050:
00051:     def create(self, validated_data):
00052:         date = validated_data.get('date')
00053:         end_dt = validated_data.get('end_dt')
00054:         if date:
00055:             if not end_dt or end_dt <= date:
00056:                 validated_data['end_dt'] = date + timedelta(days=1)
00057:
00058:         return super().create(validated_data)
00059:
00060:
```

appschedule\serializers.py

```
00061:     def validate(self, data):
00062:         is_absence = data.get('is_absence', getattr(self.instance, 'is_absence', False
))
00063:         if is_absence:
00064:             return data
00065:         # Tomar valores del instance si no están en data (caso PATCH parcial)
00066:         lot = data.get('lot', getattr(self.instance, 'lot', None))
00067:         address = data.get('address', getattr(self.instance, 'address', None))
00068:         job = data.get('job', getattr(self.instance, 'job', None))
00069:         crew = data.get('crew', getattr(self.instance, 'crew', None))
00070:         event = data.get('event', getattr(self.instance, 'event', None))
00071:         event_id = event.pk if event else None
00072:         start_at = data.get('date', getattr(self.instance, 'date', None))
00073:         end_at = data.get('end_dt', getattr(self.instance, 'end_dt', None))
00074:
00075:         if not crew:
00076:             raise ValidationError("Crew is required")
00077:
00078:         category = crew.category
00079:
00080:         if start_at and (not end_at or end_at <= start_at):
00081:             end_at = start_at + timedelta(days=1)
00082:             data['end_dt'] = end_at
00083:
00084:         if not (lot or address or job):
00085:             raise ValidationError("Address or lot/job must be provided")
00086:
00087:         qs_ed = EventDraft.objects.filter(crew__category=category)
00088:         qs_e = Event.objects.filter(crew__category=category, deleted=False)
00089:
00090:         if lot:
00091:             qs_ed = qs_ed.filter(job=job, lot=lot)
00092:             qs_e = qs_e.filter(job=job, lot=lot)
00093:         elif address:
00094:             qs_ed = qs_ed.filter(address=address)
00095:             qs_e = qs_e.filter(address=address)
00096:
00097:         if self.instance:
00098:             qs_ed = qs_ed.exclude(pk=self.instance.pk)
00099:         if event_id:
00100:             qs_ed = qs_ed.exclude(event_id=event_id)
00101:             qs_e = qs_e.exclude(pk=event_id)
00102:
00103:         if qs_ed.exists() or qs_e.exists():
00104:             raise ValidationError("Duplicate Event Detected")
00105:
00106:         return data
00107:
00108:
00109: class EventNoteSerializer(serializers.ModelSerializer):
00110:     class Meta:
00111:         model = EventNote
00112:         fields = ['notes', 'updated_at', 'updated_by', 'event']
00113:         read_only_fields = ['updated_at', 'updated_by']
00114:
00115:     def create(self, validated_data):
00116:         print('create validated_data', validated_data)
00117:         event = validated_data['event']
00118:         validated_data['updated_by'] = self.context['request'].user
00119:         print('create validated_data', validated_data)
```

appschedule\serializers.py

```
00120:         try:
00121:             event_note, created = EventNote.objects.update_or_create(
00122:                 event=event,
00123:                 defaults=validated_data
00124:             )
00125:             return event_note
00126:         except Exception as e:
00127:             raise serializers.ValidationError(f"Error creating EventNote: {e}")
00128:
00129: class UserSerializer(serializers.ModelSerializer):
00130:     class Meta:
00131:         model = User
00132:         fields = ['id', 'username']
00133:
00134:
00135: class EventChatMessageSerializer(serializers.ModelSerializer):
00136:     author = UserSerializer(read_only=True)
00137:
00138:     class Meta:
00139:         model = EventChatMessage
00140:         fields = ['id', 'event', 'author', 'message', 'timestamp']
00141:         read_only_fields = ['id', 'timestamp', 'author']
00142:         extra_kwargs = {'event': {'write_only': True}}
00143:
00144:     def create(self, validated_data):
00145:         event = validated_data.pop('event')
00146:         author = self.context['request'].user
00147:         return EventChatMessage.objects.create(event=event, author=author, **validated
_data)
00148:
00149: class AbsenceReasonSerializer(serializers.ModelSerializer):
00150:     class Meta:
00151:         model = AbsenceReason
00152:         fields = ['id', 'name', 'description']
00153:
00154:
00155: class EventImageSerializer(serializers.ModelSerializer):
00156:     image_url = serializers.SerializerMethodField()
00157:
00158:     class Meta:
00159:         model = EventImage
00160:         fields = ['id', 'event', 'image', 'image_url', 'uploaded_at', 'uploaded_by']
00161:         read_only_fields = ['id', 'uploaded_at', 'uploaded_by', 'image_url']
00162:
00163:     def get_image_url(self, obj):
00164:         request = self.context.get('request')
00165:         if obj.image and request:
00166:             return request.build_absolute_uri(obj.image.url)
00167:         elif obj.image:
00168:             return obj.image.url
00169:         return None
```

appschedule\signals.py

```
00001: import asyncio
00002: import json
00003: import os
00004:
00005: from django.db.models.signals import post_save, post_delete
00006: from django.dispatch import receiver
00007: from django.conf import settings
00008: from channels.layers import get_channel_layer
00009: from asgiref.sync import async_to_sync   # OAHP
00010: from appschedule.models import Event, EventDraft, EventNote, EventChatMessage, EventCh
atReadStatus, EventImage
00011: from appschedule.serializers import EventSerializer, EventDraftSerializer, EventNoteSe
rializer, EventChatMessageSerializer
00012:
00013:
00014: @receiver(post_save, sender=Event)
00015: def event_saved(sender, instance, **kwargs):
00016:     if getattr(settings, 'ENABLE_WEBSOCKET_NOTIFICATIONS', False):
00017:         channel_layer = get_channel_layer()
00018:         serializer = EventSerializer(instance)
00019:
00020:         event_data = serializer.data
00021:         asyncio.run(channel_layer.group_send(
00022:             "calendar_updates",
00023:             {
00024:                 'type': 'event.updated',
00025:                 'event_data': event_data,
00026:             }
00027:         ))
00028:
00029:
00030: @receiver(post_delete, sender=Event)
00031: def event_deleted(sender, instance, **kwargs):
00032:     if getattr(settings, 'ENABLE_WEBSOCKET_NOTIFICATIONS', False):
00033:         channel_layer = get_channel_layer()
00034:         event_data = {
00035:             'id': instance.id,
00036:         }
00037:         asyncio.run(channel_layer.group_send(
00038:             "calendar_updates",
00039:             {
00040:                 'type': 'event.updated',
00041:                 'event_data': event_data,
00042:             }
00043:         ))
00044:
00045:
00046: @receiver(post_save, sender=EventDraft)
00047: def event_draft_saved(sender, instance, **kwargs):
00048:     if getattr(settings, 'ENABLE_WEBSOCKET_NOTIFICATIONS', False):
00049:         channel_layer = get_channel_layer()
00050:         serializer = EventDraftSerializer(instance)
00051:
00052:         event_data = serializer.data
00053:         asyncio.run(channel_layer.group_send(
00054:             "calendar_updates",
00055:             {
00056:                 'type': 'event_draft.updated',
00057:                 'event_data': event_data,
00058:             }
```

appschedule\signals.py

```
00059:         ))
00060:
00061:
00062: @receiver(post_delete, sender=EventDraft)
00063: def event_draft_deleted(sender, instance, **kwargs):
00064:     if getattr(settings, 'ENABLE_WEBSOCKET_NOTIFICATIONS', False):
00065:         channel_layer = get_channel_layer()
00066:         event_data = {
00067:             'id': instance.id,
00068:         }
00069:         asyncio.run(channel_layer.group_send(
00070:             "calendar_updates",
00071:             {
00072:                 'type': 'event_draft.updated',
00073:                 'event_data': event_data,
00074:             }
00075:         ))
00076:
00077: @receiver(post_save, sender=EventNote)
00078: def event_note_saved(sender, instance, **kwargs):
00079:     if getattr(settings, 'ENABLE_WEBSOCKET_NOTIFICATIONS', False):
00080:         channel_layer = get_channel_layer()
00081:         serializer = EventNoteSerializer(instance)
00082:
00083:         event_data = serializer.data
00084:         asyncio.run(channel_layer.group_send(
00085:             f"event_{instance.event_id}_notes",
00086:             {
00087:                 'type': 'note.updated',
00088:                 'event_data': event_data,
00089:             }
00090:         ))
00091:
00092:
00093: @receiver(post_save, sender=EventChatMessage)
00094: def event_chatmessage_saved(sender, instance, created, **kwargs):
00095:     if not created:
00096:         return
00097:
00098:     # Crear automáticamente el ReadStatus para el autor
00099:     EventChatReadStatus.objects.update_or_create(
00100:         user=instance.author,
00101:         message=instance,
00102:         defaults={"read_at": instance.timestamp}
00103:     )
00104:
00105:     if getattr(settings, 'ENABLE_WEBSOCKET_NOTIFICATIONS', False):
00106:         channel_layer = get_channel_layer()
00107:         serializer = EventChatMessageSerializer(instance)
00108:         event_data = serializer.data
00109:
00110:         try:
00111:             asyncio.run(channel_layer.group_send(
00112:                 f"schedule_{instance.event_id}_chat",
00113:                 {
00114:                     'type': 'chat.updated',
00115:                     'data': event_data,
00116:                     'author_id': instance.author.id  # ■ Añadido aquí
00117:                 }
00118:             ))
```

appschedule\signals.py

```
00119:         except Exception as e:
00120:             print(f"[WebSocket Error] {e}")
00121:
00122:
00123: @receiver(post_delete, sender=EventImage)
00124: def delete_event_image_file(sender, instance, **kwargs):
00125:     # Borra el archivo físico cuando se elimina el registro
00126:     if instance.image:
00127:         image_path = instance.image.path
00128:         instance.image.delete(False)
00129:         # Ahora intentamos borrar la carpeta si queda vacía
00130:         import os
00131:         dir_path = os.path.dirname(image_path)
00132:         try:
00133:             # Si la carpeta está vacía, la borra
00134:             if os.path.isdir(dir_path) and not os.listdir(dir_path):
00135:                 os.rmdir(dir_path)
00136:         except Exception as e:
00137:             # Si hay error (por ejemplo, permisos), solo lo imprime, no detiene el pro
ceso
00138:             print(f"Error al borrar carpeta vacía: {dir_path} -> {e}")
```

appschedule\templates\schedule_pdf.html

```
00001: {% load custom_filters %}
00002: <!DOCTYPE html>
00003: <html lang="en">
00004: <head>
00005:   <meta charset="UTF-8" />
00006:   <title>Schedule Report</title>
00007:   <style>
00008:     @page {
00009:     size: A4 landscape;
00010:     margin: 1.0cm;
00011:   }
00012:
00013:   body {
00014:     font-family: Arial, sans-serif;
00015:     font-size: 14px; /* Aumentado */
00016:     margin: 0;
00017:     padding: 0;
00018:     color: #111; /* Más contraste */
00019:     font-weight: bold;
00020:   }
00021:
00022:   .logo {
00023:     width: 200px;
00024:     margin-bottom: 10px;
00025:     float: left;
00026:   }
00027:
00028:   .header-title {
00029:     text-align: center;
00030:     font-size: 26px; /* Aumentado */
00031:     font-weight: bold;
00032:     margin-top: 10px;
00033:     margin-bottom: 35px;
00034:     clear: both;
00035:     color: #000;
00036:     border-bottom: 2px solid #444;
00037:     padding-bottom: 8px;
00038:   }
00039:
00040:   .category-block {
00041:     page-break-before: always;
00042:   }
00043:
00044:   .category-title {
00045:     font-size: 20px;
00046:     font-weight: bold;
00047:     background-color: #dcdcdc;
00048:     padding: 10px;
00049:     margin: 0;
00050:     text-align: left;
00051:     color: #000;
00052:     border-bottom: 2px solid #555;
00053:   }
00054:
00055:   table {
00056:     width: 100%;
00057:     border-collapse: collapse;
00058:     margin-bottom: 20px;
00059:     table-layout: fixed;
00060:   }
```

appschedule\templates\schedule_pdf.html

```
00061:
00062:   th, td {
00063:     border: 1px solid #444; /* Más oscuro */
00064:     padding: 8px;
00065:     text-align: center;
00066:     font-size: 13px; /* Aumentado */
00067:     width: 100px;
00068:     height: 90px;
00069:     word-wrap: break-word;
00070:   }
00071:
00072:   th {
00073:     background-color: #e0e0e0;
00074:     font-size: 14px;
00075:     color: #000;
00076:     border-bottom: 2px solid #555;
00077:   }
00078:
00079:   td.crew {
00080:     text-align: left;
00081:     font-weight: bold;
00082:     background-color: #f3f3f3;
00083:     width: 130px;
00084:     font-size: 13px;
00085:   }
00086:
00087:   .no-events {
00088:     color: #bbb;
00089:     font-style: italic;
00090:   }
00091:
00092:   .event-entry {
00093:     margin-bottom: 6px;
00094:     font-size: 13px;
00095:     color: #000;
00096:   }
00097:
00098:   .event-entry.absence {
00099:     color: #000;
00100:     font-weight: bold;
00101:     background-color: #ffffff;
00102:     padding: 2px 4px;
00103:     border-radius: 4px;
00104:   }
00105:
00106:   .event-entry.finishing {
00107:     color: #6c757d;
00108:     font-style: italic;
00109:   }
00110:
00111:   .ext-service {
00112:     color: red;
00113:     font-weight: bold;
00114:     margin-left: 4px;
00115:   }
00116:
00117:   .extended {
00118:     color: #000;
00119:     font-weight: bold;
00120:     background-color: #ffffff;
```

appschedule\templates\schedule_pdf.html

```
00121:     border-radius: 4px;
00122:     padding: 2px 4px;
00123:   }
00124:
00125:   .finishing-up {
00126:     color: #666;
00127:     font-style: italic;
00128:   }
00129: </style>
00130:
00131: </head>
00132: <body>
00133:
00134:   {% if logo_url %}
00135:     <img src="{{ logo_url }}" class="logo" />
00136:   {% endif %}
00137:
00138:   <div class="header-title">Schedule Report: {{ date_range }}</div>
00139:
00140:   {% with 0 as counter %}
00141:     {% for category, crews in categorized_events.items %}
00142:       {% if category != '5■■ Slabs' %}
00143:         {% if counter > 0 %}
00144:           <div style="page-break-before: always;"></div>
00145:         {% endif %}
00146:         <div class="category-title">{{ category }}</div>
00147:         <table>
00148:           <thead>
00149:             <tr>
00150:               <th>Crew</th>
00151:               {% for day in days %}
00152:                 <th>{{ day|date:"l, M d" }}</th>
00153:               {% endfor %}
00154:             </tr>
00155:           </thead>
00156:           <tbody>
00157:             {% for crew_name, events_by_day in crews.items %}
00158:               <tr>
00159:                 <td class="crew">{{ crew_name }}</td>
00160:                 {% for day in days %}
00161:                   <td>
00162:                     {% with events=events_by_day|get_item:day %}
00163:                       {% if events %}
00164:                         {% for event in events %}
00165:                           {% if "■ Absence" in event|stringformat:"s" %}
00166:                             <div class="event-entry absence">{{ event }}</div>
00167:                           {% elif event|stringformat:"s" == "Finishing up work" %}
00168:                             <div class="event-entry finishing">■ Finishing up work</di
v>
00169:                           {% else %}
00170:                             <div class="event-entry{% if event.extended_service %} ext
ended{% endif %}">
00171:                               {{ event.title }}
00172:                               {% if event.extended_service %}
00173:                                 <span class="ext-service">■ Ext. Service</span>
00174:                               {% endif %}
00175:                               {% if event.description %}
00176:                                 – {{ event.description }}
00177:                               {% endif %}
00178:                             </div>
```

appschedule\templates\schedule_pdf.html

```
00179:                            {% endif %}
00180:                          {% endfor %}
00181:                        {% else %}
00182:                          <span class="no-events">–</span>
00183:                        {% endif %}
00184:                      {% endwith %}
00185:                    </td>
00186:                  {% endfor %}
00187:                </tr>
00188:              {% endfor %}
00189:            </tbody>
00190:          </table>
00191:          {% with counter|add:"1" as counter %}
00192:          {% endwith %}
00193:        {% endif %}
00194:      {% endfor %}
00195:    {% endwith %}
00196:
00197: </body>
00198: </html>
```

appschedule\templatetags\custom_filters.py

```
00001: from django import template
00002: register = template.Library()
00003:
00004: @register.filter
00005: def get_item(d, k):
00006:     try:
00007:         return d.get(k, [])
00008:     except Exception:
00009:         return []
```

appschedule\tests.py

```
00001: from django.test import TestCase
00002:
00003: # Create your tests here.
```

appschedule\urls.py

```
00001: from django.urls import path, include
00002: from rest_framework.routers import DefaultRouter
00003: from .views import (
00004:     EventViewSet, EventsListView, EventNoteViewSet,
00005:     EventChatViewSet, download_schedule_pdf, MyEventsView,
00006:     export_schedule_excel, AbsenceReasonViewSet, WeeklySupervisorStatsChartView,
00007:     WeeklySupervisorStatsExcelView, unread_chat_counts, mark_chat_read,
00008:     EventImageViewSet
00009: )
00010:
00011: router = DefaultRouter()
00012: router.register(r'schedule', EventViewSet)
00013: router.register(r'absence-reasons', AbsenceReasonViewSet)
00014: router.register(r'event-images', EventImageViewSet, basename='eventimage')
00015:
00016: urlpatterns = [
00017:     path('api/', include(router.urls)),
00018:     path('api/schedule-list/', EventsListView.as_view()),
00019:     path('api/events/<int:event_id>/note/', EventNoteViewSet.as_view({'get': 'retrieve
', 'post': 'create'}), name='event-note'),
00020:     path('api/events/<int:event_id>/chat/messages/', EventChatViewSet.as_view({'get':
'list', 'post': 'create'}), name='event-chat-messages'),
00021:     path('api/schedule-report/', download_schedule_pdf, name='download_schedule_pdf'),
00022:     path('api/my-events/', MyEventsView.as_view(), name='my_events'),
00023:     path('api/schedule-excel/', export_schedule_excel, name='export_schedule_excel'),
00024:     path('api/supervisor-stats/', WeeklySupervisorStatsChartView.as_view(), name='supe
rvisor-stats'),
00025:     path('api/supervisor-stats-excel/', WeeklySupervisorStatsExcelView.as_view()),
00026:     path('api/unread-chat-counts/', unread_chat_counts, name='unread_chat_counts'),
00027:     path('api/mark-chat-read/<int:event_id>/', mark_chat_read, name='mark_chat_read'),
00028: ]
```

appschedule\views.py

```
00001: from django_filters.rest_framework import DjangoFilterBackend
00002: from django.shortcuts import get_object_or_404
00003: from rest_framework.permissions import (
00004:     IsAuthenticated, DjangoModelPermissions,
00005:     DjangoModelPermissionsOrAnonReadOnly, IsAuthenticated
00006: )
00007: from rest_framework import viewsets, status
00008: from rest_framework.response import Response
00009: from rest_framework.authentication import TokenAuthentication
00010: from rest_framework.exceptions import ValidationError
00011: from rest_framework.decorators import action, api_view, permission_classes
00012: from django.db.models import (
00013:     Q, Count, OuterRef, Subquery, DateTimeField, ExpressionWrapper, F,
00014:     Value, IntegerField
00015: )
00016: from django.db import connection                 # OAHP
00017: from django.db.models.functions import TruncWeek, Coalesce  # OAHP <-
00018: from appschedule.models import Event, EventChatMessage, EventChatReadStatus
  # OAHP
00019: from django.utils import timezone
00020: from django.utils.timezone import now            # OAHP
00021: from asgiref.sync import async_to_sync           # OAHP
00022: from django.contrib.auth import get_user_model   # OAHP
00023: from rest_framework.views import APIView
00024: from rest_framework.pagination import PageNumberPagination
00025:
00026: from .models import (
00027:     Event, EventDraft, EventNote, EventChatMessage, Crew, AbsenceReason,
00028:     EventChatReadStatus, EventImage
00029: )
00030: from crewsapp.models import Category, Job
00031: from .serializers import (
00032:     EventSerializer, EventDraftSerializer, EventNoteSerializer, EventChatMessageSerial
izer,
00033:     AbsenceReasonSerializer, EventImageSerializer
00034: )
00035: from .filters import EventDraftFilter
00036:
00037: import base64
00038: from django.utils.dateparse import parse_date
00039: from django.template.loader import render_to_string
00040: from collections import defaultdict
00041: from datetime import timedelta, datetime
00042: from weasyprint import (
00043:     CSS,
00044:     HTML,
00045: )
00046: from weasyprint.text.fonts import FontConfiguration
00047: from channels.layers import get_channel_layer
00048:
00049: from django.http import HttpResponse
00050: import openpyxl
00051: from openpyxl import Workbook
00052: from openpyxl.utils import get_column_letter
00053: from openpyxl.styles import Font, Border, Side, Alignment, PatternFill, PatternFill
00054: # Image
00055: from rest_framework.parsers import MultiPartParser, FormParser
00056:
00057: class EventViewSet(viewsets.ModelViewSet):
00058:     """ Event ViewSet """
```

appschedule\views.py

```
00059:     queryset = EventDraft.objects.all()
00060:     serializer_class = EventDraftSerializer
00061:     filter_backends = [DjangoFilterBackend]
00062:     filterset_class = EventDraftFilter
00063:     permission_classes = [DjangoModelPermissions]
00064:
00065:     def perform_create(self, serializer):
00066:         serializer.save(created_by=self.request.user)
00067:
00068:     def perform_update(self, serializer):
00069:         serializer.save(created_by=self.request.user)
00070:
00071:     def get_queryset(self):
00072:         # if self.action == 'events_public':
00073:         #     return Event.objects.select_related('crew').all()
00074:         return EventDraft.objects.select_related('crew').all()
00075:
00076:     def _publish_draft(self, draft):
00077:         # print('draft.event_id:: ', draft.event)
00078:         if draft.event_id is not None:
00079:             event = Event.objects.get(id=draft.event_id)
00080:             # print('draft.__dict__:: ', event.__dict__)
00081:         else:
00082:             event = Event()
00083:         excluded_fields = ['event', 'updated_at', 'created_at', 'pk', 'id']
00084:         for field in EventDraft._meta.get_fields():
00085:             if field.concrete and field.name not in excluded_fields:
00086:                 try:
00087:                     setattr(event, field.name, getattr(draft, field.name))
00088:                 except AttributeError:
00089:                     # Handle cases where the Event model might not have the exact same
 field
00090:                     print(f"Warning: Event model does not have field '{field.name}'")
00091:         event.save()
00092:         draft.delete()
00093:         print('order published: ')
00094:
00095:     def create(self, request, *args, **kwargs):
00096:         to_publish = request.data.pop('_post', False)
00097:         if to_publish and not request.user.has_perm('appschedule.add_event'):
00098:             return Response({'message': 'You do not have permission to publish events'
}, status=status.HTTP_403_FORBIDDEN)
00099:         serializer = self.get_serializer(data=request.data)
00100:         serializer.is_valid(raise_exception=True)
00101:         self.perform_create(serializer)
00102:         headers = self.get_success_headers(serializer.data)
00103:
00104:         if to_publish:
00105:             draft = self.queryset.get(pk=serializer.data['id'])
00106:             self._publish_draft(draft)
00107:             return Response({'message': 'Draft published and deleted'}, status=status.
HTTP_201_CREATED,
00108:                             headers=headers)
00109:         else:
00110:             return Response(serializer.data, status=status.HTTP_201_CREATED, headers=h
eaders)
00111:
00112:     def update(self, request, *args, **kwargs):
00113:         partial = kwargs.pop('partial', False)
00114:         to_publish = request.data.pop('_post', False)
```

appschedule\views.py

```
00115:         if to_publish and not request.user.has_perm('appschedule.add_event'):
00116:             return Response({'message': 'You do not have permission to publish events'
}, status=status.HTTP_403_FORBIDDEN)
00117:         instance = self.get_object()
00118:         serializer = self.get_serializer(instance, data=request.data, partial=partial)
00119:         serializer.is_valid(raise_exception=True)
00120:         self.perform_update(serializer)
00121:
00122:         if getattr(instance, '_prefetched_objects_cache', None):
00123:             instance._prefetched_objects_cache = {}
00124:
00125:         if to_publish:
00126:             self._publish_draft(instance)
00127:             return Response({'message': 'Draft updated, published and deleted'}, statu
s=status.HTTP_200_OK)
00128:         else:
00129:             return Response(serializer.data, status=status.HTTP_200_OK)
00130:
00131:     def destroy(self, request, *args, **kwargs):
00132:         deleted = request.query_params.get('deleted', False)
00133:         if deleted:
00134:             event = Event.objects.get(pk=kwargs['pk'])
00135:             event.deleted = True
00136:             event.save()
00137:             return Response({'message': f'Event with ID {event.id} has been deleted'},
 status=status.HTTP_200_OK)
00138:         instance = self.get_object()
00139:         self.perform_destroy(instance)
00140:         return Response(status=status.HTTP_204_NO_CONTENT)
00141:
00142:     @action(detail=False, methods=['post'])
00143:     def publish_drafts(self, request):
00144:         """
00145:         Publica los eventos draft dentro del rango de fechas proporcionado.
00146:         Recibe 'start_date' y 'end_date' en el cuerpo de la petición POST.
00147:         """
00148:         start_date_str = request.data.get('start_date')
00149:         end_date_str = request.data.get('end_date')
00150:
00151:         if not start_date_str or not end_date_str:
00152:             return Response({'error': 'Start and end dates must be provided.'},
00153:                             status=status.HTTP_400_BAD_REQUEST)
00154:
00155:         try:
00156:             start_date = timezone.datetime.strptime(start_date_str, '%Y-%m-%d').date()
00157:             end_date = timezone.datetime.strptime(end_date_str, '%Y-%m-%d').date()
00158:         except ValueError:
00159:             return Response({'error': 'The date format must be YYYY-MM-DD.'},
00160:                             status=status.HTTP_400_BAD_REQUEST)
00161:
00162:         events_to_publish = EventDraft.objects.filter(
00163:             date__gte=start_date,
00164:             end_dt__lt=end_date
00165:         )
00166:         for event in events_to_publish:
00167:             self._publish_draft(event)
00168:         return Response({}, status=status.HTTP_200_OK)
00169:
00170:
00171: class EventsListView(APIView):
```

appschedule\views.py

```
00172:     permission_classes = [IsAuthenticated]
00173:
00174:     def get(self, request, format=None):
00175:         start_at = self.request.query_params.get('start_at')
00176:         end_at = self.request.query_params.get('end_at')
00177:
00178:         if not start_at:
00179:             raise ValidationError('start_at must be provided')
00180:         if not end_at:
00181:             raise ValidationError('end_at must be provided')
00182:         q = Q()
00183:         q &= (Q(date__lte=start_at,
00184:                 end_dt__gte=start_at) |  # Empieza antes o en el inicio y termina desp
ués o en el inicio
00185:               Q(date__gte=start_at, date__lt=end_at) |  # Empieza dentro del rango
00186:               Q(date__lte=start_at, end_dt__lt=end_at))  # Termina dentro del rango
00187:         q &= (Q(date__lte=end_at, end_dt__gte=end_at) |  # Empieza antes o en el fin y
 termina después o en el fin
00188:               Q(date__gte=start_at, date__lte=end_at) |  # Empieza dentro del rango
00189:               Q(end_dt__gte=start_at, end_dt__lte=end_at))  # Termina dentro del rango
00190:
00191:         events = Event.objects.select_related('crew').filter(q, deleted=False)
00192:
00193:         # Excluir los eventos que ya tienen draft (solo para usuarios con permiso)
00194:         if request.user.has_perm('appschedule.add_eventdraft'):
00195:             exclude_list = EventDraft.objects.exclude(event__isnull=True).values_list(
'event_id', flat=True)
00196:             if len(exclude_list) > 0:
00197:                 events = events.exclude(id__in=exclude_list)
00198:
00199:         serializer_event = EventSerializer(events.distinct(), many=True)
00200:         response = {
00201:             'events': serializer_event.data,
00202:         }
00203:
00204:         # Agregar drafts si el usuario tiene permiso
00205:         if request.user.has_perm('appschedule.add_eventdraft'):
00206:             drafts = EventDraft.objects.select_related('crew').filter(q).distinct()
00207:             serializer_draft = EventDraftSerializer(drafts, many=True)
00208:             response['drafts'] = serializer_draft.data
00209:
00210:         # Agregar resumen por categoría del crew
00211:         category_counts = (
00212:             events
00213:             .filter(is_absence=False)
00214:             .values('crew__category__name')
00215:             .annotate(total=Count('id'))
00216:         )
00217:         response['category_totals'] = list(category_counts)
00218:
00219:         return Response(response, status=status.HTTP_200_OK)
00220:
00221:
00222: class EventNoteViewSet(viewsets.ViewSet):
00223:     permission_classes = [IsAuthenticated]
00224:     serializer_class = EventNoteSerializer
00225:     lookup_field = 'event_id'
00226:
00227:     def retrieve(self, request, event_id=None):
00228:         try:
```

appschedule\views.py

```
00229:             event = get_object_or_404(Event, pk=event_id)
00230:             try:
00231:                 note = EventNote.objects.get(event=event)
00232:                 serializer = self.serializer_class(note)
00233:                 return Response(serializer.data)
00234:             except EventNote.DoesNotExist:
00235:                 return Response({'notes': ''}, status=status.HTTP_200_OK)  # Or 404 if
 you prefer
00236:         except Event.DoesNotExist:
00237:             return Response({'error': f'Event with ID {event_id} not found.'}, status=
status.HTTP_404_NOT_FOUND)
00238:
00239:     def create(self, request, event_id=None):
00240:         try:
00241:             get_object_or_404(Event, pk=event_id)
00242:             event_note = EventNote.objects.filter(event=event_id).first()
00243:             if event_note:
00244:                 serializer = self.serializer_class(event_note, data=request.data, cont
ext={'request': request})
00245:             else:
00246:                 serializer = self.serializer_class(data=request.data, context={'reques
t': request})
00247:             serializer.is_valid(raise_exception=True)
00248:             serializer.save()
00249:             return Response(serializer.data, status=status.HTTP_200_OK)
00250:         except Exception as e:
00251:             return Response({'error': str(e)}, status=status.HTTP_400_BAD_REQUEST)
00252:
00253:
00254: User = get_user_model()
00255:
00256: class EventChatViewSet(viewsets.ViewSet):
00257:     permission_classes = [IsAuthenticated]
00258:     serializer_class = EventChatMessageSerializer
00259:
00260:     def list(self, request, event_id=None):
00261:         if not event_id:
00262:             return Response({"error": "Event ID is required."}, status=400)
00263:
00264:         event = get_object_or_404(Event, pk=event_id)
00265:         queryset = EventChatMessage.objects.filter(event=event).order_by('timestamp')
00266:         serializer = self.serializer_class(queryset, many=True)
00267:
00268:         return Response(serializer.data)
00269:
00270:     def create(self, request, event_id=None):
00271:         if not event_id:
00272:             return Response({"error": "Event ID is required."}, status=400)
00273:
00274:         event = get_object_or_404(Event, pk=event_id)
00275:         serializer = self.serializer_class(data=request.data, context={'request': requ
est})
00276:         serializer.is_valid(raise_exception=True)
00277:         message = serializer.save(event=event)
00278:
00279:         # Notify other users via WebSocket based on latest read message
00280:         channel_layer = get_channel_layer()
00281:         other_users = User.objects.exclude(id=request.user.id)
00282:
00283:         for user in other_users:
```

appschedule\views.py

```
00284:             # Get the latest message that the user has read
00285:             last_read_entry = EventChatReadStatus.objects.filter(user=user).order_by('
-message__timestamp').first()
00286:
00287:             unread_count = (
00288:                 EventChatMessage.objects
00289:                 .filter(event=event)
00290:                 .exclude(author=user)  # No contar sus propios mensajes como no leídos
00291:             )
00292:
00293:             if last_read_entry:
00294:                 unread_count = unread_count.filter(timestamp__gt=last_read_entry.messa
ge.timestamp)
00295:
00296:             count = unread_count.count()
00297:
00298:             group_name = f"user_{user.id}_unread"
00299:             async_to_sync(channel_layer.group_send)(
00300:                 group_name,
00301:                 {
00302:                     "type": "unread.updated",
00303:                     "event_id": event.id,
00304:                     "count": count,
00305:                     "from_user_id": request.user.id,
00306:                 }
00307:             )
00308:
00309:         return Response(serializer.data, status=201)
00310:
00311:
00312:
00313: @permission_classes([IsAuthenticated])
00314: @api_view(['GET'])
00315: def download_schedule_pdf(request):
00316:     start_at = request.GET.get('start_at')
00317:     end_at = request.GET.get('end_at')
00318:     # print(f"■■ Dates: {start_at} - {end_at}")
00319:
00320:     if not start_at or not end_at:
00321:         return Response({'error': 'start_at and end_at are required'}, status=400)
00322:
00323:     start_date = parse_date(start_at)
00324:     end_date = parse_date(end_at)
00325:
00326:     # end_date se toma como tope, no inclusive
00327:     days = []
00328:     current = start_date
00329:     while current < end_date:
00330:         days.append(current)
00331:         current += timedelta(days=1)
00332:
00333:     # print("■ Days to render:")
00334:     for d in days:
00335:         d.strftime('%A, %b %d')
00336:
00337:     # Consulta eventos cruzados en el rango
00338:     q = Q()
00339:     q &= (Q(date__lte=start_date, end_dt__gte=start_date) |
00340:           Q(date__gte=start_date, date__lt=end_date) |
00341:           Q(date__lte=start_date, end_dt__lt=end_date))
```

appschedule\views.py

```
00342:     q &= (Q(date__lte=end_date, end_dt__gte=end_date) |
00343:           Q(date__gte=start_date, date__lte=end_date) |
00344:           Q(end_dt__gte=start_date, end_dt__lte=end_date))
00345:
00346:     events = Event.objects.select_related('crew', 'crew__category').filter(q, deleted=
False)
00347:
00348:     # Todos los crews activos agrupados por categoría
00349:     all_crews = Crew.objects.select_related('category').filter(status=True)
00350:     categorized_events = defaultdict(lambda: defaultdict(lambda: defaultdict(list)))
00351:
00352:     for crew in all_crews:
00353:         if crew.category:
00354:             categorized_events[crew.category.name][crew.name]  # inicializar aunque no
 tenga eventos
00355:
00356:     for event in events:
00357:         if not event.crew or not event.crew.category:
00358:             continue
00359:
00360:         crew_name = event.crew.name
00361:         category_name = event.crew.category.name
00362:
00363:         # Expande cualquier evento, marcando inicio y fin
00364:         current = event.date
00365:         while current < event.end_dt:
00366:             if start_date <= current <= end_date:
00367:                 if event.is_absence and event.absence_reason:
00368:                     absence_text = f"■ Absence: {event.absence_reason.name}"
00369:                     if event.description:
00370:                         absence_text += f" – {event.description}"
00371:                     categorized_events[category_name][crew_name][current].append(absen
ce_text)
00372:                 else:
00373:                     if current == event.date:
00374:                         categorized_events[category_name][crew_name][current].append(e
vent)
00375:                     else:
00376:                         categorized_events[category_name][crew_name][current].append("
Finishing up work")
00377:             current += timedelta(days=1)
00378:
00379:     categorized_events_clean = {
00380:         cat: {
00381:             crew: dict(days)
00382:             for crew, days in crews.items()
00383:         }
00384:         for cat, crews in categorized_events.items()
00385:     }
00386:
00387:     domain = request.get_host()
00388:     if 'phoenixelectricandair' in domain:
00389:         tenant_logo = 'media/tenant_logos/Logo-phoenix-w.png'
00390:     elif '192.168.0.248:8000' in domain or 'division16llc' in domain:
00391:         tenant_logo = 'media/tenant_logos/Logo-division-w.png'
00392:     else:
00393:         tenant_logo = 'media/tenant_logos/default-logo.png'
00394:
00395:     logo_url = request.build_absolute_uri('/' + tenant_logo)
00396:
```

appschedule\views.py

```
00397:     context = {
00398:         'categorized_events': categorized_events_clean,
00399:         'date_range': f"{start_date.strftime('%b %d')} – {(end_date - timedelta(days=1
)).strftime('%b %d, %Y')}",
00400:         'days': days,
00401:         'logo_url': logo_url
00402:     }
00403:
00404:     html = render_to_string('schedule_pdf.html', context)
00405:     font_config = FontConfiguration()
00406:     pdf_file = HTML(string=html).write_pdf(font_config=font_config)
00407:
00408:     return Response({
00409:         'file': base64.b64encode(pdf_file),
00410:         'filename': f'schedule_{start_at}_to_{end_at}.pdf',
00411:         'file_type': 'application/pdf'
00412:     }, status=200)
00413:
00414:
00415: class MyEventsView(APIView):
00416:     permission_classes = [IsAuthenticated]
00417:     pagination_class = PageNumberPagination
00418:     pagination_class.page_size = 20
00419:
00420:     def get(self, request, format=None):
00421:         search = self.request.query_params.get('search')
00422:         user = request.user
00423:
00424:         q = Q(deleted=False)
00425:
00426:         #  Detecta si el usuario es supervisor (tiene jobs asignados)
00427:         jobs = Job.objects.filter(crews__members=user).values_list('id', flat=True)
00428:
00429:         if jobs.exists():
00430:             # Solo ve sus comunidades
00431:             q &= Q(job__in=jobs)
00432:             # print(f"OjO  Usuario {user.username} es supervisor con {len(jobs)} comun
idades")
00433:         else:
00434:             # print(f"✔ Usuario {user.username} no tiene comunidades asignadas. Mostr
ando todos los eventos")
00435:
00436:             pass
00437:
00438:         queryset = Event.objects.select_related('crew', 'crew__category').filter(q).di
stinct()
00439:
00440:         # Subquery para contar mensajes NO leídos por usuario
00441:         unread_subquery = EventChatMessage.objects.filter(
00442:             event=OuterRef('pk')
00443:         ).exclude(
00444:             read_statuses__user=user
00445:         ).values('event').annotate(
00446:             count=Count('id')
00447:         ).values('count')[:1]
00448:
00449:         queryset = queryset.annotate(
00450:             unread_count=Subquery(unread_subquery, output_field=IntegerField())
00451:         ).annotate(
00452:             unread_count_fixed=Coalesce('unread_count', Value(0))
```

appschedule\views.py

```
00453:         )
00454:
00455:         queryset = queryset.order_by('-unread_count_fixed', '-date')
00456:
00457:         if search:
00458:             queryset = queryset.filter(
00459:                 Q(title__icontains=search) |
00460:                 Q(description__icontains=search) |
00461:                 Q(crew__name__icontains=search)
00462:             )
00463:
00464:         paginator = self.pagination_class()
00465:         page = paginator.paginate_queryset(queryset, request, view=self)
00466:         serializer = EventSerializer(page, many=True)
00467:         return paginator.get_paginated_response(serializer.data)
00468:
00469:
00470: @permission_classes([IsAuthenticated])
00471: @api_view(['GET'])
00472: def export_schedule_excel(request):
00473:     start_at = request.GET.get("start_at")
00474:     end_at = request.GET.get("end_at")
00475:
00476:     try:
00477:         start_date = datetime.strptime(start_at, "%Y-%m-%d").date()
00478:         end_date = datetime.strptime(end_at, "%Y-%m-%d").date()
00479:     except:
00480:         return HttpResponse("Invalid dates", status=400)
00481:
00482:     # Filtro robusto para eventos cruzados
00483:     q = Q()
00484:     q &= (Q(date__lte=start_date, end_dt__gte=start_date) |
00485:           Q(date__gte=start_date, date__lte=end_date))
00486:     q &= (Q(date__lte=end_date, end_dt__gte=end_date) |
00487:           Q(end_dt__gte=start_date, end_dt__lte=end_date))
00488:
00489:     events = Event.objects.filter(q, deleted=False).select_related("crew", "crew__cate
gory").order_by("crew__name")
00490:
00491:     # Construir días
00492:     days = [start_date + timedelta(days=i) for i in range((end_date - start_date).days
)]
00493:
00494:     # Agrupar eventos por categoría, crew y día (ajustando end_dt - 1 día)
00495:     categorized_events = defaultdict(lambda: defaultdict(lambda: defaultdict(list)))
00496:
00497:     for event in events:
00498:         category = event.crew.category.name if hasattr(event.crew, "category") and eve
nt.crew.category else "Uncategorized"
00499:         crew_name = event.crew.name
00500:         adjusted_end = event.end_dt - timedelta(days=1)  # Aquí - 1 dia
00501:         event_range = [event.date + timedelta(days=i) for i in range((adjusted_end - e
vent.date).days + 1)]
00502:         for day in days:
00503:             if day in event_range:
00504:                 info = f"{event.title}"
00505:                 if event.extended_service:
00506:                     info += " ■Ext"
00507:                 if event.description:
00508:                     info += f" - {event.description}"
```

appschedule\views.py

```
00509:                 categorized_events[category][crew_name][day].append(info)
00510:
00511:     # Crear workbook
00512:     wb = openpyxl.Workbook()
00513:     wb.remove(wb.active)  # borrar la hoja vacía por defecto
00514:
00515:     for category, crews in categorized_events.items():
00516:         ws = wb.create_sheet(title=category[:31])  # máximo 31 caracteres
00517:         header = ["Crew"] + [d.strftime("%Y-%m-%d") for d in days]
00518:         ws.append(header)
00519:
00520:         # Aplicar negrita al header
00521:         for col in range(1, len(header) + 1):
00522:             cell = ws.cell(row=1, column=col)
00523:             cell.font = Font(bold=True)
00524:             cell.alignment = Alignment(horizontal='center', vertical='center')
00525:             cell.fill = PatternFill(start_color="F3F3F3", end_color="F3F3F3", fill_typ
e="solid")
00526:
00527:         for crew_name, events_by_day in crews.items():
00528:             row = [crew_name]
00529:             for d in days:
00530:                 daily_events = events_by_day[d]
00531:                 cell_text = "\n".join(daily_events) if daily_events else ""
00532:                 row.append(cell_text)
00533:             ws.append(row)
00534:
00535:         # Ajustar estilos
00536:         for row in ws.iter_rows(min_row=2, max_row=ws.max_row):
00537:             for cell in row:
00538:                 cell.alignment = Alignment(wrap_text=True, vertical='top')
00539:
00540:         for col in range(1, len(header) + 1):
00541:             ws.column_dimensions[get_column_letter(col)].width = 20
00542:
00543:     # Devolver el archivo
00544:     response = HttpResponse(
00545:         content_type='application/vnd.openxmlformats-officedocument.spreadsheetml.shee
t',
00546:     )
00547:     filename = f"schedule_{start_date}_to_{(end_date - timedelta(days=1))}.xlsx"
00548:     print(filename)
00549:     response['Content-Disposition'] = f'attachment; filename="{filename}"'
00550:     wb.save(response)
00551:     return response
00552:
00553: class AbsenceReasonViewSet(viewsets.ModelViewSet):
00554:     queryset = AbsenceReason.objects.filter(is_active=True)
00555:     serializer_class = AbsenceReasonSerializer
00556:     authentication_classes = [TokenAuthentication]
00557:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00558:
00559:
00560: class WeeklySupervisorStatsChartView(APIView):
00561:     permission_classes = [IsAuthenticated]
00562:
00563:     def get(self, request):
00564:         # Leer parámetros del querystring
00565:         start_date_str = request.query_params.get('start_date')
00566:         end_date_str = request.query_params.get('end_date')
```

appschedule\views.py

```
00567:         category = request.query_params.get('category')
00568:
00569:         # Si no hay fechas en los query params, usar 12 semanas atrás hasta hoy
00570:         today = datetime.today().date()
00571:         try:
00572:             start_date = datetime.strptime(start_date_str, "%Y-%m-%d").date() if start
_date_str else today - timedelta(weeks=12)
00573:             end_date = datetime.strptime(end_date_str, "%Y-%m-%d").date() if end_date_
str else today
00574:         except ValueError:
00575:             return Response({"error": "Invalid date format. Use YYYY-MM-DD."}, status=
400)
00576:
00577:         # Armamos filtro dinámico y lista de parámetros
00578:         category_filter = ""
00579:         params = [start_date, end_date]
00580:
00581:         if category:
00582:             category_filter = "AND cc.name = %s"
00583:             params.append(category)
00584:
00585:         with connection.cursor() as cursor:
00586:             cursor.execute(f"""
00587:                 SELECT
00588:                     DATE_SUB(e.date, INTERVAL WEEKDAY(e.date) DAY) AS week,
00589:                     au.username AS supervisor,
00590:                     COUNT(DISTINCT e.id) AS total_events
00591:                 FROM appschedule_event e
00592:                 JOIN crewsapp_crew c ON e.crew_id = c.id
00593:                 JOIN crewsapp_crew_jobs crj ON e.job_id = crj.job_id
00594:                 JOIN crewsapp_crew_members crm ON crj.crew_id = crm.crew_id
00595:                 JOIN auth_user au ON crm.user_id = au.id
00596:                 JOIN crewsapp_category cc ON c.category_id = cc.id
00597:                 WHERE e.deleted = FALSE
00598:                   AND e.is_absence = FALSE
00599:                   AND e.date BETWEEN %s AND %s
00600:                   {category_filter}
00601:                 GROUP BY week, au.username
00602:                 ORDER BY week, au.username
00603:             """, params)
00604:
00605:             rows = cursor.fetchall()
00606:
00607:         # Procesamos los datos para Chart.js
00608:         data_map = {}
00609:         labels_set = set()
00610:
00611:         for week, supervisor, total in rows:
00612:             week_str = week.strftime('%m-%d-%Y')
00613:             labels_set.add(week_str)
00614:             if supervisor not in data_map:
00615:                 data_map[supervisor] = {}
00616:             data_map[supervisor][week_str] = total
00617:
00618:         labels = sorted(labels_set)
00619:         datasets = []
00620:
00621:         for supervisor, week_data in data_map.items():
00622:             dataset = {
00623:                 "label": supervisor,
```

appschedule\views.py

```
00624:                 "data": [week_data.get(week, 0) for week in labels]
00625:             }
00626:             datasets.append(dataset)
00627:
00628:         return Response({
00629:             "labels": labels,
00630:             "datasets": datasets
00631:         })
00632:
00633:
00634: class WeeklySupervisorStatsExcelView(APIView):
00635:     permission_classes = [IsAuthenticated]
00636:
00637:     def get(self, request):
00638:         # Rango por semanas
00639:         range_weeks = int(request.query_params.get('weeks', 16))
00640:         today = datetime.today().date()
00641:
00642:         # Fechas de filtro
00643:         start_date_str = request.query_params.get('start_date')
00644:         end_date_str = request.query_params.get('end_date')
00645:         category = request.query_params.get('category')  # Filtro dinámico
00646:
00647:         try:
00648:             start_date = datetime.strptime(start_date_str, "%Y-%m-%d").date() if start
_date_str else today - timedelta(weeks=range_weeks)
00649:             end_date = datetime.strptime(end_date_str, "%Y-%m-%d").date() if end_date_
str else today
00650:         except ValueError:
00651:             return Response({"error": "Invalid date format. Use YYYY-MM-DD."}, status=
400)
00652:
00653:         # Agregar filtro de categoría si se envía
00654:         category_filter = ""
00655:         params = [start_date, end_date]
00656:         if category:
00657:             category_filter = "AND cc.name = %s"
00658:             params.append(category)
00659:
00660:         with connection.cursor() as cursor:
00661:             cursor.execute(f"""
00662:                 SELECT
00663:                     cc.name AS category,
00664:                     DATE_SUB(e.date, INTERVAL WEEKDAY(e.date) DAY) AS week,
00665:                     au.username AS supervisor,
00666:                     COUNT(DISTINCT e.id) AS total_events
00667:                 FROM appschedule_event e
00668:                 JOIN crewsapp_crew c ON e.crew_id = c.id
00669:                 JOIN crewsapp_crew_jobs crj ON e.job_id = crj.job_id
00670:                 JOIN crewsapp_crew_members crm ON crj.crew_id = crm.crew_id
00671:                 JOIN auth_user au ON crm.user_id = au.id
00672:                 JOIN crewsapp_category cc ON c.category_id = cc.id
00673:                 WHERE e.deleted = FALSE
00674:                   AND e.is_absence = FALSE
00675:                   AND e.date BETWEEN %s AND %s
00676:                   {category_filter}
00677:                 GROUP BY cc.name, week, au.username
00678:                 ORDER BY cc.name, week, au.username
00679:             """, params)
00680:
```

appschedule\views.py

```
00681:             rows = cursor.fetchall()
00682:
00683:         # Agrupación y estructuración por categoría
00684:         from collections import defaultdict
00685:         categorized_data = defaultdict(lambda: defaultdict(dict))
00686:         weeks_per_category = defaultdict(set)
00687:
00688:         for category, week, supervisor, total in rows:
00689:             week_str = week.strftime('%Y-%m-%d')
00690:             categorized_data[category][supervisor][week_str] = total
00691:             weeks_per_category[category].add(week_str)
00692:
00693:         # Crear archivo Excel
00694:         wb = Workbook()
00695:         wb.remove(wb.active)
00696:
00697:         bold = Font(bold=True)
00698:         center = Alignment(horizontal="center", vertical="center")
00699:         border = Border(
00700:             left=Side(style="thin"), right=Side(style="thin"),
00701:             top=Side(style="thin"), bottom=Side(style="thin")
00702:         )
00703:
00704:         for category, supervisor_data in categorized_data.items():
00705:             ws = wb.create_sheet(title=category[:31])
00706:             sorted_weeks = sorted(weeks_per_category[category])
00707:             supervisors = sorted(supervisor_data.keys())
00708:
00709:             # Header
00710:             ws.append(["Week"] + supervisors)
00711:             for cell in ws[1]:
00712:                 cell.font = bold
00713:                 cell.alignment = center
00714:                 cell.border = border
00715:
00716:             # Data por semana
00717:             for week in sorted_weeks:
00718:                 row = [week]
00719:                 for supervisor in supervisors:
00720:                     row.append(supervisor_data[supervisor].get(week, 0))
00721:                 ws.append(row)
00722:
00723:             # Fila separadora
00724:             sep_row = ["" for _ in range(len(supervisors) + 1)]
00725:             ws.append(sep_row)
00726:
00727:             # Totales por supervisor
00728:             total_row = ["TOTAL"]
00729:             for supervisor in supervisors:
00730:                 total = sum(supervisor_data[supervisor].values())
00731:                 total_row.append(total)
00732:             ws.append(total_row)
00733:
00734:             # Estilo a la fila TOTAL
00735:             last_row_idx = ws.max_row
00736:             for cell in ws[last_row_idx]:
00737:                 cell.font = bold
00738:                 cell.alignment = center
00739:                 cell.border = border
00740:
```

appschedule\views.py

```
00741:             # Ancho de columnas
00742:             for col in ws.columns:
00743:                 max_len = max(len(str(cell.value or "")) for cell in col)
00744:                 col_letter = col[0].column_letter
00745:                 ws.column_dimensions[col_letter].width = max_len + 2
00746:
00747:         # Retorno de archivo como respuesta HTTP
00748:         response = HttpResponse(
00749:             content_type="application/vnd.openxmlformats-officedocument.spreadsheetml.
sheet"
00750:         )
00751:         filename = f"supervisor_report_{today}.xlsx"
00752:         response["Content-Disposition"] = f'attachment; filename="{filename}"'
00753:         wb.save(response)
00754:         return response
00755:
00756:
00757: @api_view(['GET'])
00758: @permission_classes([IsAuthenticated])
00759: def unread_chat_counts(request):
00760:     user = request.user
00761:
00762:     # Buscamos trabajos (comunidades) vinculados a este usuario
00763:     jobs = Job.objects.filter(crews__members=user).values_list('id', flat=True)
00764:
00765:     if jobs.exists():
00766:         # Es supervisor → limitar a sus comunidades
00767:         event_queryset = Event.objects.filter(deleted=False, job__in=jobs)
00768:     else:
00769:         # No tiene comunidades asignadas → ve todos los eventos activos
00770:         event_queryset = Event.objects.filter(deleted=False)
00771:
00772:     # print(f"OjO  Usuario {user.username} tiene acceso a {len(jobs)} comunidades")
00773:
00774:     unread_counts = (
00775:         EventChatMessage.objects
00776:         .filter(event__in=event_queryset)
00777:         .exclude(read_statuses__user=user)
00778:         .values('event')
00779:         .annotate(count=Count('id'))
00780:     )
00781:
00782:     result = {item['event']: item['count'] for item in unread_counts}
00783:     return Response(result)
00784:
00785:
00786: @api_view(['POST'])
00787: @permission_classes([IsAuthenticated])
00788: def mark_chat_read(request, event_id):
00789:     user = request.user
00790:     event = get_object_or_404(Event, pk=event_id)
00791:
00792:     unread_messages = EventChatMessage.objects.filter(
00793:         event=event
00794:     ).exclude(read_statuses__user=user)
00795:
00796:     for msg in unread_messages:
00797:         EventChatReadStatus.objects.get_or_create(user=user, message=msg)
00798:
00799:     return Response({"status": "read updated"})
```

appschedule\views.py

```
00800:
00801:
00802: class EventImageViewSet(viewsets.ModelViewSet):
00803:     queryset = EventImage.objects.all()
00804:     serializer_class = EventImageSerializer
00805:     parser_classes = [MultiPartParser, FormParser]
00806:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00807:
00808:     def perform_create(self, serializer):
00809:         serializer.save(uploaded_by=self.request.user)
00810:
00811:     def get_queryset(self):
00812:         event_id = self.request.query_params.get('event')
00813:         qs = super().get_queryset()
00814:         if event_id:
00815:             qs = qs.filter(event_id=event_id)
00816:         return qs
00817:
00818:     @action(detail=False, methods=['post'], url_path='upload', parser_classes=[MultiPa
rtParser, FormParser])
00819:     def upload_images(self, request):
00820:         event_id = request.data.get('event_id')
00821:         if not event_id:
00822:             return Response({'error': 'Missing event_id'}, status=status.HTTP_400_BAD_
REQUEST)
00823:
00824:         try:
00825:             event = Event.objects.get(id=event_id)
00826:         except Event.DoesNotExist:
00827:             return Response({'error': 'Event not found'}, status=status.HTTP_404_NOT_F
OUND)
00828:
00829:         images = request.FILES.getlist('images')
00830:         if not images:
00831:             return Response({'error': 'No images uploaded'}, status=status.HTTP_400_BA
D_REQUEST)
00832:
00833:         created_images = []
00834:         for img in images:
00835:             instance = EventImage.objects.create(event=event, image=img, uploaded_by=r
equest.user)
00836:             created_images.append(instance)
00837:
00838:         serializer = EventImageSerializer(created_images, many=True, context={'request
': request})
00839:         return Response(serializer.data, status=status.HTTP_201_CREATED)
00840:
00841:
00842:     @action(detail=True, methods=['get'])
00843:     def images(self, request, pk=None):
00844:         event = self.get_object()
00845:         images = event.images.all()
00846:         serializer = EventImageSerializer(images, many=True, context={'request': reque
st})
00847:         return Response(serializer.data)
```

apptransactions__init__.py

```
00001:
```

apptransactions\admin.py

```
00001: from django.contrib import admin
00002: from django.forms.models import BaseInlineFormSet
00003: from django.core.exceptions import ValidationError
00004: from .models import (
00005:     Party, PartyType, PartyCategory,
00006:     DocumentType, Document, DocumentLine
00007: )
00008:
00009: @admin.register(PartyType)
00010: class PartyTypeAdmin(admin.ModelAdmin):
00011:     list_display = ('name', 'description', 'is_active')
00012:     list_filter = ('is_active',)
00013:     search_fields = ('name', 'description')
00014:     ordering = ('name',)
00015:
00016: @admin.register(PartyCategory)
00017: class PartyCategoryAdmin(admin.ModelAdmin):
00018:     list_display = ('name', 'description', 'is_active')
00019:     list_filter = ('is_active',)
00020:     search_fields = ('name', 'description')
00021:     ordering = ('name',)
00022:
00023: @admin.register(Party)
00024: class PartyAdmin(admin.ModelAdmin):
00025:     list_display = ('name', 'category', 'default_price_type', 'customer_rank', 'suppli
er_rank', 'is_active')
00026:     list_filter = ('is_active', 'category')
00027:     search_fields = ('name', 'rfc', 'email', 'phone')
00028:     autocomplete_fields = ('category', 'default_price_type', 'types')
00029:
00030:     def get_queryset(self, request):
00031:         return super().get_queryset(request).annotate()
00032:
00033:     def get_list_filter(self, request):
00034:         filters = list(super().get_list_filter(request))
00035:         filters.append(('customer_rank', admin.BooleanFieldListFilter))
00036:         filters.append(('supplier_rank', admin.BooleanFieldListFilter))
00037:         return filters
00038:
00039:
00040: @admin.register(DocumentType)
00041: class DocumentTypeAdmin(admin.ModelAdmin):
00042:     list_display = ('id', 'type_code', 'description', 'stock_movement', 'is_active')
00043:     search_fields = ('type_code', 'description')
00044:     list_filter = ('is_active', 'stock_movement')
00045:
00046: # Validación que aplica a cada línea del inline
00047: class DocumentLineInlineFormSet(BaseInlineFormSet):
00048:     def clean(self):
00049:         super().clean()
00050:         for form in self.forms:
00051:             if not form.cleaned_data.get('DELETE', False):
00052:                 quantity = form.cleaned_data.get('quantity')
00053:                 product = form.cleaned_data.get('product')
00054:                 if quantity is None or product is None:
00055:                     raise ValidationError("Todas las líneas deben tener producto y can
tidad.")
00056:
00057: class DocumentLineInline(admin.TabularInline):
00058:     model = DocumentLine
```

apptransactions\admin.py

```
00059:     extra = 1
00060:     formset = DocumentLineInlineFormSet
00061:
00062:     def get_formset(self, request, obj=None, **kwargs):
00063:         formset = super().get_formset(request, obj, **kwargs)
00064:         for field in ['quantity', 'product']:
00065:             if field in formset.form.base_fields:
00066:                 formset.form.base_fields[field].required = True  # Campos obligatorios
00067:         return formset
00068:
00069: @admin.register(Document)
00070: class DocumentAdmin(admin.ModelAdmin):
00071:     list_display = ('document_type', 'date', 'party', 'warehouse', 'is_active')
00072:     search_fields = ('job', 'lot', 'notes')
00073:     list_filter = ('document_type', 'warehouse', 'is_active')
00074:     autocomplete_fields = ('party', 'warehouse', 'document_type', 'created_by')
00075:     inlines = [DocumentLineInline]
00076:
00077:     def get_formset(self, request, obj=None, **kwargs):
00078:         formset = super().get_formset(request, obj, **kwargs)
00079:         for field in ['quantity', 'product']:
00080:             if field in formset.form.base_fields:
00081:                 formset.form.base_fields[field].required = True
00082:         return formset
```

apptransactions\apps.py

```
00001: from django.apps import AppConfig
00002:
00003:
00004: class ApptransactionsConfig(AppConfig):
00005:     default_auto_field = 'django.db.models.BigAutoField'
00006:     name = 'apptransactions'
00007:
00008:     def ready(self):
00009:         import apptransactions.signals
```

apptransactions\migrations\0001_initial.py

```
00001: # Generated by Django 5.0.3 on 2025-04-10 03:16
00002:
00003: import django.db.models.deletion
00004: from django.conf import settings
00005: from django.db import migrations, models
00006:
00007:
00008: class Migration(migrations.Migration):
00009:
00010:     initial = True
00011:
00012:     dependencies = [
00013:         ('appinventory', '0001_initial'),
00014:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00015:     ]
00016:
00017:     operations = [
00018:         migrations.CreateModel(
00019:             name='DocumentType',
00020:             fields=[
00021:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00022:                 ('type_code', models.CharField(max_length=20, unique=True)),
00023:                 ('description', models.CharField(max_length=255)),
00024:                 ('affects_physical', models.BooleanField(default=True)),
00025:                 ('affects_logical', models.BooleanField(default=True)),
00026:                 ('affects_accounting', models.BooleanField(default=False)),
00027:                 ('is_taxable', models.BooleanField(default=False)),
00028:                 ('is_purchase', models.BooleanField(default=False)),
00029:                 ('is_sales', models.BooleanField(default=False)),
00030:                 ('warehouse_required', models.BooleanField(default=True)),
00031:                 ('stock_movement', models.SmallIntegerField(choices=[(1, '+1 Entrada')
, (-1, '-1 Salida'), (0, '0 Neutro')], default=0)),
00032:                 ('is_active', models.BooleanField(default=True)),
00033:             ],
00034:         ),
00035:         migrations.CreateModel(
00036:             name='PartyCategory',
00037:             fields=[
00038:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00039:                 ('name', models.CharField(max_length=100)),
00040:                 ('is_active', models.BooleanField(default=True)),
00041:             ],
00042:         ),
00043:         migrations.CreateModel(
00044:             name='PartyType',
00045:             fields=[
00046:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00047:                 ('name', models.CharField(max_length=50)),
00048:                 ('is_active', models.BooleanField(default=True)),
00049:             ],
00050:         ),
00051:         migrations.CreateModel(
00052:             name='Document',
00053:             fields=[
00054:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00055:                 ('date', models.DateTimeField(auto_now_add=True)),
```

apptransactions\migrations\0001_initial.py

```
00056:                 ('builder', models.CharField(blank=True, max_length=100, null=True)),
00057:                 ('job', models.CharField(blank=True, max_length=100, null=True)),
00058:                 ('lot', models.CharField(blank=True, max_length=100, null=True)),
00059:                 ('notes', models.TextField(blank=True)),
00060:                 ('total_discount', models.DecimalField(decimal_places=2, default=0, ma
x_digits=10)),
00061:                 ('total_amount', models.DecimalField(decimal_places=2, default=0, max_
digits=12)),
00062:                 ('is_active', models.BooleanField(default=True)),
00063:                 ('created_by', models.ForeignKey(null=True, on_delete=django.db.models
.deletion.SET_NULL, to=settings.AUTH_USER_MODEL)),
00064:                 ('warehouse', models.ForeignKey(null=True, on_delete=django.db.models.
deletion.SET_NULL, to='appinventory.warehouse')),
00065:                 ('document_type', models.ForeignKey(on_delete=django.db.models.deletio
n.CASCADE, to='apptransactions.documenttype')),
00066:             ],
00067:         ),
00068:         migrations.CreateModel(
00069:             name='DocumentLine',
00070:             fields=[
00071:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00072:                 ('quantity', models.DecimalField(decimal_places=2, max_digits=10)),
00073:                 ('unit_price', models.DecimalField(decimal_places=2, max_digits=10)),
00074:                 ('discount_percentage', models.DecimalField(decimal_places=2, default=
0, max_digits=5)),
00075:                 ('final_price', models.DecimalField(blank=True, decimal_places=2, max_
digits=10, null=True)),
00076:                 ('brand', models.ForeignKey(blank=True, null=True, on_delete=django.db
.models.deletion.SET_NULL, to='appinventory.productbrand')),
00077:                 ('document', models.ForeignKey(on_delete=django.db.models.deletion.CAS
CADE, related_name='lines', to='apptransactions.document')),
00078:                 ('price_type', models.ForeignKey(blank=True, null=True, on_delete=djan
go.db.models.deletion.SET_NULL, to='appinventory.pricetype')),
00079:                 ('product', models.ForeignKey(on_delete=django.db.models.deletion.CASC
ADE, to='appinventory.product')),
00080:                 ('unit', models.ForeignKey(null=True, on_delete=django.db.models.delet
ion.SET_NULL, to='appinventory.unitofmeasure')),
00081:                 ('warehouse', models.ForeignKey(blank=True, null=True, on_delete=djang
o.db.models.deletion.SET_NULL, to='appinventory.warehouse')),
00082:             ],
00083:         ),
00084:         migrations.CreateModel(
00085:             name='Party',
00086:             fields=[
00087:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00088:                 ('name', models.CharField(max_length=255)),
00089:                 ('rfc', models.CharField(blank=True, max_length=50)),
00090:                 ('street', models.CharField(blank=True, max_length=100)),
00091:                 ('floor_office', models.CharField(blank=True, max_length=100)),
00092:                 ('city', models.CharField(blank=True, max_length=100)),
00093:                 ('state', models.CharField(blank=True, max_length=100)),
00094:                 ('zipcode', models.CharField(blank=True, max_length=20)),
00095:                 ('country', models.CharField(blank=True, max_length=100)),
00096:                 ('phone', models.CharField(blank=True, max_length=20)),
00097:                 ('email', models.EmailField(blank=True, max_length=254)),
00098:                 ('customer_rank', models.PositiveIntegerField(default=0)),
00099:                 ('supplier_rank', models.PositiveIntegerField(default=0)),
00100:                 ('is_active', models.BooleanField(default=True)),
```

apptransactions\migrations\0001_initial.py

```
00101:                 ('default_price_type', models.ForeignKey(blank=True, null=True, on_del
ete=django.db.models.deletion.SET_NULL, to='appinventory.pricetype')),
00102:                 ('category', models.ForeignKey(null=True, on_delete=django.db.models.d
eletion.SET_NULL, to='apptransactions.partycategory')),
00103:                 ('types', models.ManyToManyField(to='apptransactions.partytype')),
00104:             ],
00105:         ),
00106:         migrations.AddField(
00107:             model_name='document',
00108:             name='party',
00109:             field=models.ForeignKey(null=True, on_delete=django.db.models.deletion.SET
_NULL, to='apptransactions.party'),
00110:         ),
00111:     ]
```

apptransactions\migrations\0002_alter_partytype_options_partytype_description_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-08-19 00:02
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('apptransactions', '0001_initial'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterModelOptions(
00014:             name='partytype',
00015:             options={'ordering': ['name'], 'verbose_name': 'Party Type', 'verbose_name
_plural': 'Party Types'},
00016:         ),
00017:         migrations.AddField(
00018:             model_name='partytype',
00019:             name='description',
00020:             field=models.TextField(blank=True),
00021:         ),
00022:         migrations.AlterField(
00023:             model_name='partytype',
00024:             name='name',
00025:             field=models.CharField(max_length=150, unique=True),
00026:         ),
00027:     ]
```

apptransactions\migrations\0003_alter_documenttype_options_alter_partytype_options.py

```
00001: # Generated by Django 5.0.3 on 2025-08-19 22:44
00002:
00003: from django.db import migrations
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('apptransactions', '0002_alter_partytype_options_partytype_description_and_mo
re'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterModelOptions(
00014:             name='documenttype',
00015:             options={'ordering': ['-id'], 'verbose_name': 'Document Type', 'verbose_na
me_plural': 'Document Types'},
00016:         ),
00017:         migrations.AlterModelOptions(
00018:             name='partytype',
00019:             options={'ordering': ['-id'], 'verbose_name': 'Party Type', 'verbose_name_
plural': 'Party Types'},
00020:         ),
00021:     ]
```

apptransactions\migrations\0004_partycategory_description.py

```
00001: # Generated by Django 5.0.3 on 2025-08-19 23:19
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('apptransactions', '0003_alter_documenttype_options_alter_partytype_options')
,
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AddField(
00014:             model_name='partycategory',
00015:             name='description',
00016:             field=models.TextField(blank=True),
00017:         ),
00018:     ]
```

apptransactions\migrations\0005_alter_partycategory_options_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-08-20 00:00
00002:
00003: import django.db.models.functions.text
00004: from django.db import migrations, models
00005:
00006:
00007: class Migration(migrations.Migration):
00008:
00009:     dependencies = [
00010:         ('apptransactions', '0004_partycategory_description'),
00011:     ]
00012:
00013:     operations = [
00014:         migrations.AlterModelOptions(
00015:             name='partycategory',
00016:             options={'ordering': ['name']},
00017:         ),
00018:         migrations.AddConstraint(
00019:             model_name='partycategory',
00020:             constraint=models.UniqueConstraint(django.db.models.functions.text.Lower('
name'), name='uq_partycategory_name_ci'),
00021:         ),
00022:     ]
```

apptransactions\migrations\0006_alter_partycategory_options_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-08-20 03:21
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('apptransactions', '0005_alter_partycategory_options_and_more'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterModelOptions(
00014:             name='partycategory',
00015:             options={'ordering': ['-id'], 'verbose_name': 'Party Category', 'verbose_n
ame_plural': 'Party Categories'},
00016:         ),
00017:         migrations.RemoveConstraint(
00018:             model_name='partycategory',
00019:             name='uq_partycategory_name_ci',
00020:         ),
00021:         migrations.AlterField(
00022:             model_name='partycategory',
00023:             name='name',
00024:             field=models.CharField(max_length=150, unique=True),
00025:         ),
00026:     ]
```

apptransactions\migrations__init__.py

```
00001:
```

apptransactions\models.py

```
00001: # App: transactions (documentos, detalles, clientes, proveedores)
00002:
00003: from django.db import models
00004: from django.db.models import UniqueConstraint
00005: from django.db.models.functions import Lower
00006: from django.conf import settings
00007: from django.core.exceptions import ValidationError
00008: from django.contrib.auth import get_user_model
00009: from appinventory.models import Product, UnitOfMeasure, Warehouse, PriceType, ProductB
rand
00010:
00011: User = get_user_model()
00012:
00013: class PartyType(models.Model):
00014:     name = models.CharField(max_length=150, unique=True)
00015:     description = models.TextField(blank=True)  # optional
00016:     is_active = models.BooleanField(default=True)
00017:
00018:     class Meta:
00019:         verbose_name = "Party Type"
00020:         verbose_name_plural = "Party Types"
00021:         ordering = ["-id"]
00022:
00023:     def __str__(self):
00024:         return self.name
00025:
00026: class PartyCategory(models.Model):
00027:     name = models.CharField(max_length=150, unique=True)
00028:     description = models.TextField(blank=True)
00029:     is_active = models.BooleanField(default=True)
00030:
00031:     class Meta:
00032:         verbose_name = "Party Category"
00033:         verbose_name_plural = "Party Categories"
00034:         ordering = ["-id"]
00035:
00036:     def __str__(self):
00037:         return self.name
00038:
00039: class Party(models.Model):
00040:     name = models.CharField(max_length=255, unique=True)
00041:     rfc = models.CharField(max_length=50, blank=True)
00042:     street = models.CharField(max_length=100, blank=True)
00043:     floor_office = models.CharField(max_length=100, blank=True)
00044:     city = models.CharField(max_length=100, blank=True)
00045:     state = models.CharField(max_length=100, blank=True)
00046:     zipcode = models.CharField(max_length=20, blank=True)
00047:     country = models.CharField(max_length=100, blank=True)
00048:     phone = models.CharField(max_length=20, blank=True)
00049:     email = models.EmailField(blank=True)
00050:     types = models.ManyToManyField(PartyType)
00051:     category = models.ForeignKey(PartyCategory, on_delete=models.SET_NULL, null=True)
00052:     default_price_type = models.ForeignKey(PriceType, on_delete=models.SET_NULL, null=
True, blank=True)
00053:     customer_rank = models.PositiveIntegerField(default=0)
00054:     supplier_rank = models.PositiveIntegerField(default=0)
00055:     is_active = models.BooleanField(default=True)
00056:
00057:     def is_customer(self):
00058:         return self.customer_rank > 0
```

apptransactions\models.py

```
00059:
00060:     def is_supplier(self):
00061:         return self.supplier_rank > 0
00062:
00063:     def is_both(self):
00064:         return self.customer_rank > 0 and self.supplier_rank > 0
00065:
00066:     def __str__(self):
00067:         return self.name
00068:
00069: class DocumentType(models.Model):
00070:     type_code = models.CharField(max_length=20, unique=True) # Ej: INCOME, ADJUSTMENT_
OUT, PICK
00071:     description = models.CharField(max_length=255)
00072:     affects_physical = models.BooleanField(default=True)  # INVFIS
00073:     affects_logical = models.BooleanField(default=True)   # INVLOG
00074:     affects_accounting = models.BooleanField(default=False)  # INVCON
00075:     is_taxable = models.BooleanField(default=False)       # IVA
00076:     is_purchase = models.BooleanField(default=False)      # LIBCOM
00077:     is_sales = models.BooleanField(default=False)         # LIBVTA
00078:     warehouse_required = models.BooleanField(default=True)  # ALMACE
00079:     stock_movement = models.SmallIntegerField(
00080:         choices=[(1, "+1 Entrada"), (-1, "-1 Salida"), (0, "0 Neutro")],
00081:         default=0
00082:     )
00083:     is_active = models.BooleanField(default=True)
00084:
00085:     class Meta:
00086:         ordering = ["type_code"]
00087:         verbose_name = "Document Type"
00088:         verbose_name_plural = "Document Types"
00089:
00090:     def __str__(self):
00091:         return f"{self.type_code} - {self.description}"
00092:
00093: class Document(models.Model):
00094:     document_type = models.ForeignKey(DocumentType, on_delete=models.CASCADE)
00095:     date = models.DateTimeField(auto_now_add=True)
00096:     warehouse = models.ForeignKey(Warehouse, on_delete=models.PROTECT, null=True)
00097:     party = models.ForeignKey(Party, on_delete=models.PROTECT, null=True)
00098:     builder = models.CharField(max_length=100, blank=True, null=True)
00099:     job = models.CharField(max_length=100, blank=True, null=True)
00100:     lot = models.CharField(max_length=100, blank=True, null=True)
00101:     created_by = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.SET_NULL
, null=True)
00102:     notes = models.TextField(blank=True)
00103:     total_discount = models.DecimalField(max_digits=10, decimal_places=2, default=0)
00104:     total_amount = models.DecimalField(max_digits=12, decimal_places=2, default=0)
00105:     is_active = models.BooleanField(default=True)
00106:
00107:     def clean(self):
00108:         if not hasattr(self, "document_type") or self.document_type is None:
00109:             return  # Detener validación si no se ha asignado aún
00110:
00111:         if self.document_type and self.document_type.is_sales and not self.party.is_cu
stomer():
00112:             raise ValidationError("Selected party is not a customer.")
00113:         if self.document_type and self.document_type.is_purchase and not self.party.is
_supplier():
00114:             raise ValidationError("Selected party is not a supplier.")
```

apptransactions\models.py

```
00115:
00116:     def calculate_totals(self):
00117:         total = 0
00118:         total_discount = 0
00119:         for line in self.lines.all():
00120:             total += line.final_price or 0
00121:             discount = line.unit_price * line.quantity * (line.discount_percentage / 1
00)
00122:             total_discount += discount
00123:         self.total_amount = total
00124:         self.total_discount = total_discount
00125:         self.save()
00126:
00127: def __str__(self):
00128:     if getattr(self, "document_type", None) and self.date:
00129:         return f"{self.document_type.type_code} - {self.date}"
00130:     return "Document"
00131:
00132: class DocumentLine(models.Model):
00133:     document = models.ForeignKey(Document, related_name="lines", on_delete=models.PROT
ECT)
00134:     product = models.ForeignKey(Product, on_delete=models.PROTECT)
00135:     quantity = models.DecimalField(max_digits=10, decimal_places=2)
00136:     unit = models.ForeignKey(UnitOfMeasure, on_delete=models.PROTECT, null=True)
00137:     unit_price = models.DecimalField(max_digits=10, decimal_places=2)
00138:     discount_percentage = models.DecimalField(max_digits=5, decimal_places=2, default=
0)
00139:     final_price = models.DecimalField(max_digits=10, decimal_places=2, blank=True, nul
l=True)
00140:     warehouse = models.ForeignKey(Warehouse, on_delete=models.PROTECT, null=True, blan
k=True)
00141:     price_type = models.ForeignKey(PriceType, on_delete=models.PROTECT, null=True, bla
nk=True)
00142:     brand = models.ForeignKey(ProductBrand, on_delete=models.PROTECT, null=True, blank
=True)  # Marca específica usada en esta línea, útil para trazabilidad
00143:
00144:     def clean(self):
00145:         errors = {}
00146:         if self.quantity is None:
00147:             errors['quantity'] = 'La cantidad no puede estar vacía.'
00148:         if self.product is None:
00149:             errors['product'] = 'Debe seleccionar un producto.'
00150:         if errors:
00151:             raise ValidationError(errors)
00152:
00153:     def save(self, *args, **kwargs):
00154:         print(" 1 ■ apptransactions\models.py -> DocumentLine: def save(self, *args, *
*kwargs).")
00155:
00156:         discount = self.discount_percentage / 100
00157:         adjusted_price = self.unit_price
00158:
00159:         # El precio se mantiene tal cual viene del formulario (por unidad seleccionada
)
00160:         # No se aplica conversión aquí — solo se usa la unidad seleccionada
00161:
00162:         self.final_price = adjusted_price * self.quantity * (1 - discount)
00163:
00164:         # Si el usuario no seleccionó el tipo de precio, se usa el default del cliente
 o proveedor.
```

apptransactions\models.py

```
00165:         if not self.price_type and self.document and self.document.party and self.docu
ment.party.default_price_type:
00166:             self.price_type = self.document.party.default_price_type
00167:
00168:         super().save(*args, **kwargs)
00169:
00170:         # Recalcular totales del documento
00171:         if self.document:
00172:             self.document.calculate_totals()
00173:
00174:     def __str__(self):
00175:         unit_code = self.unit.code if self.unit else "unit"
00176:         return f"{self.product.name} x {self.quantity} {unit_code}" if self.product el
se "Detail"
00177:
00178:
```

apptransactions\serializers.py

```
00001:
00002: from rest_framework import serializers
00003: from rest_framework.validators import UniqueValidator
00004: from django.db import transaction
00005: from .models import DocumentType, PartyType, PartyCategory, Party
00006:
00007: class DocumentTypeSerializer(serializers.ModelSerializer):
00008:     class Meta:
00009:         model = DocumentType
00010:         fields = '__all__'
00011:
00012: class PartyTypeSerializer(serializers.ModelSerializer):
00013:     class Meta:
00014:         model = PartyType
00015:         fields = '__all__'
00016:
00017: class PartyCategorySerializer(serializers.ModelSerializer):
00018:     class Meta:
00019:         model = PartyCategory
00020:         fields = '__all__'
00021:
00022:
00023: from rest_framework import serializers
00024: from rest_framework.validators import UniqueValidator
00025: from django.db import transaction
00026: from .models import Party, PartyType, PartyCategory, PriceType
00027:
00028: class PartySerializer(serializers.ModelSerializer):
00029:     # IDs de relaciones (simple y performante para list/create/update)
00030:     types = serializers.PrimaryKeyRelatedField(
00031:         queryset=PartyType.objects.all(), many=True
00032:     )
00033:     category = serializers.PrimaryKeyRelatedField(
00034:         queryset=PartyCategory.objects.all(), allow_null=True, required=False
00035:     )
00036:     default_price_type = serializers.PrimaryKeyRelatedField(
00037:         queryset=PriceType.objects.all(), allow_null=True, required=False
00038:     )
00039:
00040:     # Unicidad por nombre (case-insensitive)
00041:     name = serializers.CharField(
00042:         max_length=255,
00043:         validators=[
00044:             UniqueValidator(
00045:                 queryset=Party.objects.all(),
00046:                 lookup='iexact',
00047:                 message='A party with this name already exists.'
00048:             )
00049:         ]
00050:     )
00051:
00052:     class Meta:
00053:         model = Party
00054:         fields = [
00055:             'id', 'name', 'rfc', 'street', 'floor_office', 'city', 'state',
00056:             'zipcode', 'country', 'phone', 'email',
00057:             'types', 'category', 'default_price_type',
00058:             'customer_rank', 'supplier_rank', 'is_active'
00059:         ]
00060:
```

apptransactions\serializers.py

```
00061:     # Validaciones y normalización
00062:     def validate(self, attrs):
00063:         # trim strings
00064:         for k in ['name','rfc','street','floor_office','city','state','zipcode','count
ry','phone','email']:
00065:             if k in attrs and isinstance(attrs[k], str):
00066:                 attrs[k] = attrs[k].strip()
00067:
00068:         # email en minúsculas
00069:         if attrs.get('email'):
00070:             attrs['email'] = attrs['email'].lower()
00071:
00072:         # (Opcional) política: al menos un rol cliente/proveedor
00073:         # if attrs.get('customer_rank', 0) == 0 and attrs.get('supplier_rank', 0) == 0
:
00074:         #     raise serializers.ValidationError({'customer_rank': 'Must be > 0 if not
supplier.', 'supplier_rank': 'Must be > 0 if not customer.'})
00075:
00076:         return attrs
00077:
00078:     @transaction.atomic
00079:     def create(self, validated_data):
00080:         types = validated_data.pop('types', [])
00081:         instance = Party.objects.create(**validated_data)
00082:         if types:
00083:             instance.types.set(types)
00084:         return instance
00085:
00086:     @transaction.atomic
00087:     def update(self, instance, validated_data):
00088:         types = validated_data.pop('types', None)
00089:         for attr, value in validated_data.items():
00090:             setattr(instance, attr, value)
00091:         instance.save()
00092:         if types is not None:
00093:             instance.types.set(types)
00094:         return instance
```

apptransactions\signals.py

```
00001: """
00002: Sistema de Señales para Gestión Automática de Inventario
00003:
00004: Este módulo implementa señales de Django que automatizan la sincronización
00005: entre transacciones de documentos y movimientos de inventario.
00006:
00007: Funcionalidades: oahp
00008: - Crear/actualizar movimientos de inventario al guardar líneas de documento
00009: - Eliminar movimientos de inventario al eliminar líneas de documento
00010: - Manejo automático de entradas/salidas de productos en almacenes
00011: - Consistencia de datos mediante transacciones atómicas
00012:
00013: Autor: Sistema Chalan-Pro
00014: """
00015:
00016: from decimal import Decimal
00017: from django.db.models.signals import post_save, post_delete
00018: from django.dispatch import receiver
00019: from django.db import transaction
00020: from apptransactions.models import DocumentLine
00021: from appinventory.models import InventoryMovement
00022:
00023:
00024: @receiver(post_save, sender=DocumentLine, dispatch_uid="docline_create_inventory_movem
ent")
00025: def create_inventory_movement(sender, instance, **kwargs):
00026:     print("  2 ■ apptransactions\\signals.py -> create_inventory_movement()")
00027:
00028:     def handle_movement():
00029:         try:
00030:             warehouse = instance.warehouse or instance.document.warehouse
00031:             doc_type = instance.document.document_type
00032:             movement_type = doc_type.stock_movement
00033:
00034:             if not warehouse or movement_type == 0:
00035:                 print("■ No se crea movimiento: documento neutro o sin almacén.")
00036:                 return
00037:
00038:             print(f"■ Preparando movimiento para línea {instance.id} | Producto: {inst
ance.product} | Cantidad: {instance.quantity}")
00039:
00040:             # Revisa si ya existe un movimiento para esa línea
00041:             movement = InventoryMovement.objects.filter(line_id=instance.id).first()
00042:
00043:             if movement:
00044:                 print(f"■■ Actualizando movimiento existente para línea {instance.id}
")
00045:                 movement.product = instance.product
00046:                 movement.warehouse = warehouse
00047:                 movement.quantity = instance.quantity
00048:                 movement.movement_type = movement_type
00049:                 movement.unit = instance.unit
00050:                 movement.reason = f"{doc_type.description} #{instance.document.id}"
00051:                 movement.document = str(instance.document.id)
00052:                 movement.created_by = instance.document.created_by
00053:             else:
00054:                 print(f"■ Creando nuevo movimiento para línea {instance.id}")
00055:                 movement = InventoryMovement(
00056:                     line_id=instance.id,
00057:                     product=instance.product,
```

apptransactions\signals.py

```
00058:                     warehouse=warehouse,
00059:                     quantity=instance.quantity,
00060:                     movement_type=movement_type,
00061:                     unit=instance.unit,
00062:                     reason=f"{doc_type.description} #{instance.document.id}",
00063:                     document=str(instance.document.id),
00064:                     created_by=instance.document.created_by
00065:                 )
00066:
00067:             movement.save()
00068:             print(f"■ Movimiento de inventario guardado para línea {instance.id}")
00069:
00070:         except Exception as e:
00071:             print(f"■ Error en handle_movement(): {e}")
00072:
00073:     transaction.on_commit(handle_movement)
00074:
00075:
00076: @receiver(post_delete, sender=DocumentLine, dispatch_uid="docline_delete_inventory_mov
ement")
00077: def delete_inventory_movement(sender, instance, **kwargs):
00078:     def handle_deletion():
00079:         try:
00080:             InventoryMovement.objects.filter(line_id=instance.id).delete()
00081:             print(f"■■ Movimiento eliminado para línea {instance.id}")
00082:         except Exception as e:
00083:             print(f"■ Error al eliminar movimiento para línea {instance.id}: {e}")
00084:
00085:     transaction.on_commit(handle_deletion)
```

apptransactions\tests__init__.py

```
00001:
```

apptransactions\tests\test_signals.py

```
00001: from django.test import TestCase
00002: from django.contrib.auth import get_user_model
00003: from apptransactions.models import Document, DocumentLine, DocumentType
00004: from appinventory.models import Product, Warehouse, InventoryMovement, Stock
00005: from decimal import Decimal
00006:
00007: User = get_user_model()
00008:
00009: class InventorySignalTests(TestCase):
00010:     def setUp(self):
00011:         self.user = User.objects.create(username='testuser')
00012:         self.warehouse = Warehouse.objects.create(name='Main Warehouse')
00013:         self.product = Product.objects.create(name='Widget A')
00014:         self.doc_type = DocumentType.objects.create(
00015:             name='Entrada', stock_movement=1, description="Entrada de inventario")
00016:         self.document = Document.objects.create(
00017:             document_type=self.doc_type,
00018:             created_by=self.user,
00019:             warehouse=self.warehouse
00020:         )
00021:
00022:     def test_inventory_movement_created_on_documentline_save(self):
00023:         line = DocumentLine.objects.create(
00024:             document=self.document,
00025:             product=self.product,
00026:             quantity=10,
00027:             unit=None,
00028:             warehouse=self.warehouse
00029:         )
00030:         movement = InventoryMovement.objects.filter(line_id=line.id).first()
00031:         self.assertIsNotNone(movement, "■ InventoryMovement debe haberse creado por el
 signal")
00032:         self.assertEqual(movement.quantity, Decimal('10'))
00033:
00034:     def test_inventory_movement_updates_stock_correctly(self):
00035:         line = DocumentLine.objects.create(
00036:             document=self.document,
00037:             product=self.product,
00038:             quantity=10,
00039:             unit=None,
00040:             warehouse=self.warehouse
00041:         )
00042:         stock = Stock.objects.get(product=self.product, warehouse=self.warehouse)
00043:         self.assertEqual(stock.quantity, Decimal('10'))
00044:
00045:         # Cambiar la cantidad
00046:         line.quantity = 5
00047:         line.save()
00048:
00049:         stock.refresh_from_db()
00050:         self.assertEqual(stock.quantity, Decimal('5'))
00051:
00052:     def test_inventory_movement_deleted_on_documentline_delete(self):
00053:         line = DocumentLine.objects.create(
00054:             document=self.document,
00055:             product=self.product,
00056:             quantity=10,
00057:             unit=None,
00058:             warehouse=self.warehouse
00059:         )
```

apptransactions\tests\test_signals.py

```
00060:         line.delete()
00061:
00062:         movement = InventoryMovement.objects.filter(line_id=line.id).first()
00063:         self.assertIsNone(movement, "■■ InventoryMovement debe eliminarse cuando se b
orra DocumentLine")
00064:
00065:         stock = Stock.objects.get(product=self.product, warehouse=self.warehouse)
00066:         self.assertEqual(stock.quantity, Decimal('0'))
```

apptransactions\urls.py

```
00001: from django.urls import path, include
00002: from rest_framework.routers import DefaultRouter
00003: from .views import (
00004:     DocumentTypeViewSet, PartyTypeViewSet, PartyCategoryViewSet, PartyViewSet
00005: )
00006:
00007: router = DefaultRouter()
00008: router.register(r'document-types', DocumentTypeViewSet)
00009: router.register(r'party-types', PartyTypeViewSet)
00010: router.register(r'party-categories', PartyCategoryViewSet)
00011: router.register(r'parties', PartyViewSet)
00012:
00013: urlpatterns = [
00014:     path('api/', include(router.urls)),
00015: ]
```

apptransactions\views.py

```
00001: from django.shortcuts import render
00002: from django.db import IntegrityError
00003: from rest_framework import viewsets, status
00004: from rest_framework.response import Response
00005: from django_filters.rest_framework import DjangoFilterBackend
00006: from rest_framework.filters import SearchFilter, OrderingFilter
00007: from rest_framework.permissions import IsAuthenticated, DjangoModelPermissions
00008: from .models import (
00009:     DocumentType, PartyType, PartyCategory, Party
00010: )
00011: from .serializers import (
00012:     DocumentTypeSerializer,PartyTypeSerializer,PartyCategorySerializer,PartySerializer
00013: )
00014: from rest_framework.authentication import TokenAuthentication
00015:
00016:
00017: class DocumentTypeViewSet(viewsets.ModelViewSet):
00018:     queryset = DocumentType.objects.all()
00019:     serializer_class = DocumentTypeSerializer
00020:     authentication_classes = [TokenAuthentication]
00021:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00022:
00023: class PartyTypeViewSet(viewsets.ModelViewSet):
00024:     queryset = PartyType.objects.all()
00025:     serializer_class = PartyTypeSerializer
00026:     authentication_classes = [TokenAuthentication]
00027:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00028:
00029: class PartyCategoryViewSet(viewsets.ModelViewSet):
00030:     queryset = PartyCategory.objects.all()
00031:     serializer_class = PartyCategorySerializer
00032:     authentication_classes = [TokenAuthentication]
00033:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00034:
00035: class PartyViewSet(viewsets.ModelViewSet):
00036:     queryset = Party.objects.all()
00037:     serializer_class = PartySerializer
00038:
00039:     # Filtros / búsqueda / orden
00040:     filter_backends = [DjangoFilterBackend, SearchFilter, OrderingFilter]
00041:     filterset_fields = ['is_active', 'category', 'types', 'customer_rank', 'supplier_r
ank']
00042:     search_fields = ['name', 'rfc', 'email', 'phone', 'city', 'state']
00043:     ordering_fields = ['name', 'customer_rank', 'supplier_rank', 'id']
00044:     ordering = ['name']
```

auditapp__init__.py

```
00001:
```

auditapp\admin.py

```
00001: from django.contrib import admin
00002: from .models import UserActionLog
00003:
00004: # admin.site.register(UserActionLog)
00005:
00006: class UserActionLogAdmin(admin.ModelAdmin):
00007:     list_display = ('user', 'action', 'model_name', 'object_id', 'action_time', 'full_
log')  # Add new column
00008:     search_fields = ('user__username', 'action', 'model_name', 'object_id', 'action_ti
me')  # Enables search
00009:     list_filter = ('action', 'user', 'action_time')  # Filters for easier navigation
00010:     ordering = ('-action_time',)  # Sort by most recent activity first
00011:     date_hierarchy = 'action_time'  # Adds a date-based filter in Django Admin
00012:
00013:     # Custom method to display concatenated log
00014:     def full_log(self, obj):
00015:         return f"{obj.user.username} {obj.action} {obj.model_name} {obj.object_id} on
{obj.action_time}"
00016:
00017:     full_log.short_description = "Full Log"  # Custom column title in the admin
00018:
00019: admin.site.register(UserActionLog, UserActionLogAdmin)
```

auditapp\apps.py

```
00001: from django.apps import AppConfig
00002:
00003:
00004: class AuditappConfig(AppConfig):
00005:     default_auto_field = 'django.db.models.BigAutoField'
00006:     name = 'auditapp'
00007:     verbose_name = 'Audit'
```

auditapp\migrations\0001_initial.py

```
00001: # Generated by Django 5.0.3 on 2024-10-15 04:00
00002:
00003: import django.db.models.deletion
00004: from django.conf import settings
00005: from django.db import migrations, models
00006:
00007:
00008: class Migration(migrations.Migration):
00009:
00010:     initial = True
00011:
00012:     dependencies = [
00013:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00014:     ]
00015:
00016:     operations = [
00017:         migrations.CreateModel(
00018:             name='UserActionLog',
00019:             fields=[
00020:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00021:                 ('action', models.CharField(choices=[('create', 'Create'), ('update',
'Update'), ('delete', 'Delete'), ('view', 'View')], max_length=10, verbose_name='Action Type'
)),
00022:                 ('model_name', models.CharField(max_length=255, verbose_name='Model Af
fected')),
00023:                 ('object_id', models.CharField(max_length=255, null=True, verbose_name
='Object ID')),
00024:                 ('action_time', models.DateTimeField(auto_now_add=True, verbose_name='
Action Time')),
00025:                 ('details', models.TextField(blank=True, null=True, verbose_name='Deta
ils')),
00026:                 ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE
, to=settings.AUTH_USER_MODEL, verbose_name='User')),
00027:             ],
00028:         ),
00029:     ]
```

auditapp\migrations\0002_alter_useractionlog_object_id.py

```
00001: # Generated by Django 5.0.3 on 2024-10-13 22:56
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('auditapp', '0001_initial'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterField(
00014:             model_name='useractionlog',
00015:             name='object_id',
00016:             field=models.CharField(max_length=255, null=True, verbose_name='Object ID'
),
00017:         ),
00018:     ]
```

auditapp\migrations\0003_alter_useractionlog_action.py

```
00001: # Generated by Django 5.0.3 on 2025-05-26 05:12
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('auditapp', '0002_alter_useractionlog_object_id'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterField(
00014:             model_name='useractionlog',
00015:             name='action',
00016:             field=models.CharField(max_length=10, verbose_name='Action Type'),
00017:         ),
00018:     ]
```

auditapp\migrations__init__.py

```
00001:
```

auditapp\models.py

```
00001: from django.db import models
00002: from django.contrib.auth.models import User
00003:
00004: class UserActionLog(models.Model):
00005:     ACTION_CHOICES = [
00006:         ('create', 'Create'),
00007:         ('update', 'Update'),
00008:         ('delete', 'Delete'),
00009:         ('view', 'View'),
00010:     ]
00011:
00012:     user = models.ForeignKey(User, on_delete=models.CASCADE, verbose_name="User")
00013:     # action = models.CharField(max_length=10, choices=ACTION_CHOICES, verbose_name="A
ction Type")
00014:     action = models.CharField(max_length=10, verbose_name="Action Type")
00015:     model_name = models.CharField(max_length=255, verbose_name="Model Affected")
00016:     object_id = models.CharField(null=True, max_length=255, verbose_name="Object ID")
00017:     action_time = models.DateTimeField(auto_now_add=True, verbose_name="Action Time")
00018:     details = models.TextField(null=True, blank=True, verbose_name="Details")
00019:
00020:     def __str__(self):
00021:         return f"{self.user.username} {self.action} {self.model_name} {self.object_id}
 on {self.action_time}"
```

auditapp\tests.py

```
00001: from django.test import TestCase
00002:
00003: # Create your tests here.
```

auditapp\urls.py

```
00001: from django.urls import path
00002: from .views import LogUserActionView
00003: 
00004: urlpatterns = [
00005:     path('api/log-action/', LogUserActionView.as_view(), name='log_user_action'),
00006: ]
```

auditapp\views.py

```
00001: from django.views.decorators.csrf import csrf_exempt
00002: from rest_framework.views import APIView
00003: from rest_framework.response import Response
00004: from rest_framework.permissions import IsAuthenticated
00005: from .models import UserActionLog
00006: from django.utils.decorators import method_decorator
00007:
00008: @method_decorator(csrf_exempt, name='dispatch')  # Deshabilita CSRF para esta vista
00009:
00010: class LogUserActionView(APIView):
00011:     permission_classes = [IsAuthenticated]
00012:
00013:     def post(self, request):
00014:         user = request.user
00015:         action = request.data.get('action')
00016:         model_name = request.data.get('model_name')
00017:         object_id = request.data.get('object_id')
00018:         details = request.data.get('details', '')
00019:
00020:         UserActionLog.objects.create(
00021:             user=user,
00022:             action=action,
00023:             model_name=model_name,
00024:             object_id=object_id,
00025:             details=details
00026:         )
00027:
00028:         return Response({"message": "Action logged successfully"}, status=201)
00029:
```

crewsapp__init__.py

```
00001:
```

crewsapp\admin.py

```
00001: from django.contrib import admin
00002: from .models import Truck, Crew, TruckAssignment, Category
00003: 
00004: @admin.register(Truck)
00005: class TruckAdmin(admin.ModelAdmin):
00006:     list_display = ['plate_number', 'model', 'year', 'status']
00007:     list_filter = ('model', 'year', 'status')
00008:     search_fields = ['plate_number', 'model', 'year', 'status']
00009: 
00010: 
00011: @admin.register(Crew)
00012: class CrewAdmin(admin.ModelAdmin):
00013:     filter_horizontal = ['members', 'jobs']
00014:     list_display = ['name', 'category', 'status', 'permission_create_event']
00015:     list_filter = ['category']
00016:     search_fields = ['name']
00017: 
00018: admin.site.register(Category)
00019: # admin.site.register(Crew)
00020: admin.site.register(TruckAssignment)
```

crewsapp\apps.py

```
00001: from django.apps import AppConfig
00002:
00003:
00004: class CrewsConfig(AppConfig):
00005:     default_auto_field = 'django.db.models.BigAutoField'
00006:     name = 'crewsapp'
00007:     verbose_name = 'Crews and Fleet'
```

crewsapp\migrations\0001_initial.py

```
00001: # Generated by Django 5.0.3 on 2024-10-15 03:59
00002:
00003: import django.db.models.deletion
00004: from django.conf import settings
00005: from django.db import migrations, models
00006:
00007:
00008: class Migration(migrations.Migration):
00009:
00010:     initial = True
00011:
00012:     dependencies = [
00013:         ('ctrctsapp', '0001_initial'),
00014:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00015:     ]
00016:
00017:     operations = [
00018:         migrations.CreateModel(
00019:             name='Truck',
00020:             fields=[
00021:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00022:                 ('plate_number', models.CharField(max_length=20, unique=True, verbose_
name='Plate Number')),
00023:                 ('model', models.CharField(max_length=255, verbose_name='Model')),
00024:                 ('year', models.IntegerField(verbose_name='Year')),
00025:                 ('status', models.BooleanField(default=True, verbose_name='Active')),
00026:             ],
00027:             options={
00028:                 'verbose_name': 'Truck',
00029:                 'verbose_name_plural': 'Trucks',
00030:             },
00031:         ),
00032:         migrations.CreateModel(
00033:             name='Crew',
00034:             fields=[
00035:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00036:                 ('name', models.CharField(max_length=255, verbose_name='Crew Name')),
00037:                 ('status', models.BooleanField(default=True, verbose_name='Active')),
00038:                 ('jobs', models.ManyToManyField(related_name='crews', to='ctrctsapp.jo
b', verbose_name='Assigned Jobs')),
00039:                 ('members', models.ManyToManyField(related_name='crews', to=settings.A
UTH_USER_MODEL, verbose_name='Crew Members')),
00040:             ],
00041:             options={
00042:                 'verbose_name': 'Crew',
00043:                 'verbose_name_plural': 'Crews',
00044:             },
00045:         ),
00046:         migrations.CreateModel(
00047:             name='TruckAssignment',
00048:             fields=[
00049:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00050:                 ('assigned_at', models.DateTimeField(verbose_name='Assigned At')),
00051:                 ('unassigned_at', models.DateTimeField(blank=True, null=True, verbose_
name='Unassigned At')),
00052:                 ('crew', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE
, to='crewsapp.crew', verbose_name='Assigned Crew')),
```

crewsapp\migrations\0001_initial.py

```
00053:                 ('truck', models.ForeignKey(on_delete=django.db.models.deletion.CASCAD
E, to='crewsapp.truck', verbose_name='Assigned Truck')),
00054:             ],
00055:         ),
00056:     ]
```

crewsapp\migrations\0002_alter_truckassignment_assigned_at_and_more.py

```
00001: # Generated by Django 5.0.3 on 2024-10-13 12:39
00002:
00003: import django.db.models.deletion
00004: from django.db import migrations, models
00005:
00006:
00007: class Migration(migrations.Migration):
00008:
00009:     dependencies = [
00010:         ('crewsapp', '0001_initial'),
00011:     ]
00012:
00013:     operations = [
00014:         migrations.AlterField(
00015:             model_name='truckassignment',
00016:             name='assigned_at',
00017:             field=models.DateTimeField(verbose_name='Assigned At'),
00018:         ),
00019:         migrations.AlterField(
00020:             model_name='truckassignment',
00021:             name='crew',
00022:             field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='c
rewsapp.crew', verbose_name='Assigned Crew'),
00023:         ),
00024:         migrations.AlterField(
00025:             model_name='truckassignment',
00026:             name='truck',
00027:             field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='c
rewsapp.truck', verbose_name='Assigned Truck'),
00028:         ),
00029:     ]
```

crewsapp\migrations\0003_category_alter_crew_jobs_alter_crew_members_and_more.py

```
00001: # Generated by Django 5.0.3 on 2025-05-26 05:12
00002:
00003: import django.db.models.deletion
00004: from django.conf import settings
00005: from django.db import migrations, models
00006:
00007:
00008: class Migration(migrations.Migration):
00009:
00010:     dependencies = [
00011:         ('crewsapp', '0002_alter_truckassignment_assigned_at_and_more'),
00012:         ('ctrctsapp', '0012_contract_doc_type_contract_needs_reprint'),
00013:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00014:     ]
00015:
00016:     operations = [
00017:         migrations.CreateModel(
00018:             name='Category',
00019:             fields=[
00020:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00021:                 ('name', models.CharField(max_length=100, unique=True)),
00022:             ],
00023:             options={
00024:                 'verbose_name': 'Category',
00025:                 'verbose_name_plural': 'Categories',
00026:             },
00027:         ),
00028:         migrations.AlterField(
00029:             model_name='crew',
00030:             name='jobs',
00031:             field=models.ManyToManyField(blank=True, related_name='crews', to='ctrctsa
pp.job', verbose_name='Assigned Jobs'),
00032:         ),
00033:         migrations.AlterField(
00034:             model_name='crew',
00035:             name='members',
00036:             field=models.ManyToManyField(blank=True, related_name='crews', to=settings
.AUTH_USER_MODEL, verbose_name='Crew Members'),
00037:         ),
00038:         migrations.AddField(
00039:             model_name='crew',
00040:             name='category',
00041:             field=models.ForeignKey(blank=True, null=True, on_delete=django.db.models.
deletion.SET_NULL, to='crewsapp.category', verbose_name='Category'),
00042:         ),
00043:     ]
```

crewsapp\migrations\0004_crew_permission_create_event.py

```
00001: # Generated by Django 5.0.3 on 2025-07-28 22:57
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('crewsapp', '0003_category_alter_crew_jobs_alter_crew_members_and_more'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AddField(
00014:             model_name='crew',
00015:             name='permission_create_event',
00016:             field=models.BooleanField(default=False, verbose_name='Can Create/Update E
vents?'),
00017:         ),
00018:     ]
```

crewsapp\migrations\0005_alter_crew_permission_create_event.py

```
00001: # Generated by Django 5.0.3 on 2025-08-19 23:19
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('crewsapp', '0004_crew_permission_create_event'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterField(
00014:             model_name='crew',
00015:             name='permission_create_event',
00016:             field=models.BooleanField(default=False, verbose_name='Can Create/Update S
chedule?'),
00017:         ),
00018:     ]
```

crewsapp\migrations__init__.py

```
00001:
```

crewsapp\models.py

```
00001: from django.db import models
00002: from django.contrib.auth.models import User
00003: from ctrctsapp.models import Job
00004:
00005: class Category (models.Model):
00006:     name = models.CharField(max_length=100, unique=True)
00007:
00008:     def __str__(self):
00009:         return self.name
00010:
00011:     class Meta:
00012:         verbose_name = 'Category'
00013:         verbose_name_plural = 'Categories'
00014:
00015:
00016: class Truck(models.Model):
00017:     plate_number = models.CharField(max_length=20, unique=True, verbose_name='Plate Nu
mber')
00018:     model = models.CharField(max_length=255, verbose_name='Model')
00019:     year = models.IntegerField(verbose_name='Year')
00020:     status = models.BooleanField(default=True, verbose_name='Active')  # Campo para in
dicar si está activa o inactiva
00021:
00022:     class Meta:
00023:         verbose_name = 'Truck'
00024:         verbose_name_plural = 'Trucks'
00025:
00026:     def __str__(self):
00027:         return f"{self.model} - {self.plate_number}"
00028:
00029:
00030: class Crew(models.Model):
00031:     name = models.CharField(max_length=255, verbose_name='Crew Name')
00032:     members = models.ManyToManyField(User, related_name='crews', verbose_name='Crew Me
mbers', blank=True)
00033:     jobs = models.ManyToManyField(Job, related_name='crews', verbose_name='Assigned Jo
bs', blank=True)
00034:     status = models.BooleanField(default=True, verbose_name='Active')  # Campo para in
dicar si está activa o inactiva
00035:     category = models.ForeignKey(Category, on_delete=models.SET_NULL, verbose_name='Ca
tegory', null=True, blank=True)
00036:     permission_create_event = models.BooleanField(default=False, verbose_name='Can Cre
ate/Update Schedule?')
00037:
00038:     class Meta:
00039:         verbose_name = 'Crew'
00040:         verbose_name_plural = 'Crews'
00041:
00042:     def __str__(self):
00043:         return self.name
00044:
00045: class TruckAssignment(models.Model):
00046:     crew = models.ForeignKey(Crew, on_delete=models.CASCADE, verbose_name='Assigned Cr
ew')
00047:     truck = models.ForeignKey(Truck, on_delete=models.CASCADE, verbose_name='Assigned
Truck')
00048:     assigned_at = models.DateTimeField(auto_now_add=False, verbose_name='Assigned At')
00049:     unassigned_at = models.DateTimeField(auto_now_add=False, null=True, blank=True, ve
rbose_name='Unassigned At')
00050:
```

crewsapp\models.py

```
00051:     def __str__(self):
00052:         return f"{self.truck} assigned to {self.crew}"
00053:
```

crewsapp\serializers.py

```
00001: from rest_framework import serializers
00002: from .models import Crew, Truck, TruckAssignment, Category
00003: from django.contrib.auth.models import User
00004:
00005: class UserSerializer(serializers.ModelSerializer):
00006:     class Meta:
00007:         model = User
00008:         fields = ['id', 'username']
00009:
00010:
00011: class CategorySerializer(serializers.ModelSerializer):
00012:     class Meta:
00013:         model = Category
00014:         fields = '__all__'
00015:
00016:
00017: # Serializador para Truck
00018: class TruckSerializer(serializers.ModelSerializer):
00019:     class Meta:
00020:         model = Truck
00021:         fields = '__all__'
00022:
00023:
00024: # Serializador para Crew
00025: class CrewSerializer(serializers.ModelSerializer):
00026:     category_name = serializers.SerializerMethodField()
00027:
00028:     def get_category_name(self, obj):
00029:         return obj.category.name if obj.category else None
00030:
00031:     class Meta:
00032:         model = Crew
00033:         fields = ['id', 'name', 'category_name', 'status']
00034:         # depth = 1  # Para incluir las relaciones (Job, Members y Truck)
00035:
00036:
00037: class TruckAssignmentSerializer(serializers.ModelSerializer):
00038:     class Meta:
00039:         model = TruckAssignment
00040:         fields = '__all__'
```

crewsapp\tests.py

```
00001: from django.test import TestCase
00002:
00003: # Create your tests here.
```

crewsapp\urls.py

```
00001: from django.urls import path, include
00002: from rest_framework.routers import DefaultRouter
00003: from .views import (
00004:     CrewViewSet, TruckViewSet, CategoryListView, CategoryListView,
00005:     SupervisorCommunitiesView, CrewPermissionView
00006: )
00007: 
00008: router = DefaultRouter()
00009: router.register(r'crews', CrewViewSet)
00010: router.register(r'trucks', TruckViewSet)
00011: 
00012: urlpatterns = [
00013:     path('', include(router.urls)),
00014:     path('api/categories/', CategoryListView.as_view(), name='category-list'),
00015:     path('api/supervisor-communities/', SupervisorCommunitiesView.as_view(), name='sup
ervisor-communities'),
00016:     path('api/crew/supervisor/', CrewPermissionView.as_view(), name='crew-permission-p
rofile'),
00017: 
00018: ]
```

crewsapp\views.py

```
00001: from rest_framework import viewsets
00002: from rest_framework.response import Response
00003: from rest_framework.permissions import IsAuthenticated
00004: from rest_framework.views import APIView
00005: from django.db import connection
00006: from .models import Crew, Truck, TruckAssignment, Category
00007: from .serializers import (
00008:     CrewSerializer, TruckSerializer, TruckAssignmentSerializer
00009: )
00010:
00011:
00012: # ViewSet para Truck
00013: class TruckViewSet(viewsets.ModelViewSet):
00014:     queryset = Truck.objects.all()
00015:     serializer_class = TruckSerializer
00016:     permission_classes = [IsAuthenticated]
00017:
00018:
00019: # ViewSet para Crew
00020: # Filtrado para los Resources del schedule.
00021: class CrewViewSet(viewsets.ModelViewSet):
00022:     queryset = Crew.objects.filter(category__isnull=False, status=True).order_by('id')
00023:     serializer_class = CrewSerializer
00024:     permission_classes = [IsAuthenticated]
00025:
00026:
00027: # ViewSet para TruckAssignment
00028: class TruckAssignmentViewSet(viewsets.ModelViewSet):
00029:     queryset = TruckAssignment.objects.all()
00030:     serializer_class = TruckAssignmentSerializer
00031:
00032:
00033: class CategoryListView(APIView):
00034:     permission_classes = [IsAuthenticated]
00035:
00036:     def get(self, request):
00037:         categories = Category.objects.values('id', 'name')
00038:         return Response(categories)
00039:
00040:
00041: class SupervisorCommunitiesView(APIView):
00042:     def get(self, request):
00043:         query = """
00044:         SELECT
00045:             au.username AS supervisor,
00046:             j.name AS community_job
00047:         FROM crewsapp_crew_members crm
00048:         JOIN auth_user au ON crm.user_id = au.id
00049:         JOIN crewsapp_crew_jobs crj ON crm.crew_id = crj.crew_id
00050:         JOIN ctrctsapp_job j ON crj.job_id = j.id
00051:         GROUP BY au.username, j.name
00052:         ORDER BY au.username, j.name;
00053:         """
00054:
00055:         with connection.cursor() as cursor:
00056:             cursor.execute(query)
00057:             rows = cursor.fetchall()
00058:
00059:         result = {}
00060:         for supervisor, community in rows:
```

crewsapp\views.py

```
00061:             result.setdefault(supervisor, []).append(community)
00062:
00063:         return Response(result)
00064:
00065:
00066: class CrewPermissionView(APIView):
00067:     permission_classes = [IsAuthenticated]
00068:
00069:     def get(self, request):
00070:         user = request.user
00071:         crew = Crew.objects.filter(members=user).first()
00072:
00073:         is_coordinator = crew is None
00074:         can_create_event = crew.permission_create_event if crew else False
00075:
00076:         return Response({
00077:             "username": user.username,
00078:             "crew": {
00079:                 "id": crew.id,
00080:                 "name": crew.name,
00081:                 "category": {
00082:                     "id": crew.category.id,
00083:                     "name": crew.category.name
00084:                 } if crew.category else None,
00085:                 "permission_create_event": crew.permission_create_event,
00086:                 "members": list(crew.members.values_list('id', flat=True))
00087:             } if crew else None,
00088:             "can_create_event": can_create_event,
00089:             "is_coordinator": is_coordinator
00090:         })
```

ctrctsapp__init__.py

```
00001:
```

ctrctsapp\admin.py

```
00001: # admin.py
00002:
00003: from django.contrib import admin
00004: from .models import WorkPrice, Builder, Job, HouseModel, Contract, ContractDetails
00005:
00006: admin.site.site_header = "Chalan-Pro Administration"
00007: admin.site.site_title = "Chalan-Pro Admin"
00008: admin.site.index_title = "Welcome to the Chalan-Pro Admin Panel"
00009:
00010: @admin.register(WorkPrice)
00011: class WorkPriceAdmin(admin.ModelAdmin):
00012:     list_display = ['name', 'trim', 'rough', 'unit_price']
00013:     list_filter = ('builders',)
00014:     search_fields = ['name', 'trim', 'rough']
00015:
00016: @admin.register(Builder)
00017: class BuilderAdmin(admin.ModelAdmin):
00018:     list_display = ['name', 'trim_amount', 'rough_amount', 'travel_price_amount']
00019:     list_filter = ('jobs',)
00020:     search_fields = ['name']
00021:
00022: @admin.register(Job)
00023: class JobAdmin(admin.ModelAdmin):
00024:     list_display = ['name', 'builder']
00025:     list_filter = ('builder',)
00026:     search_fields = ['name', 'id']
00027:
00028: @admin.register(HouseModel)
00029: class HouseModelAdmin(admin.ModelAdmin):
00030:     list_display = ['id', 'name']
00031:     list_filter = ('jobs',)
00032:     search_fields = ['name']
00033:
00034: @admin.register(Contract)
00035: class ContractAdmin(admin.ModelAdmin):
00036:     list_display = ['id', 'type', 'builder', 'job', 'house_model', 'lot', 'sqft', 'add
ress', 'total', 'date_created']
00037:     list_filter = ['type', 'builder', 'job', 'date_created']
00038:     search_fields = ['house_model__name', 'builder__name', 'job__name', 'address', 'ty
pe']
00039:
00040: @admin.register(ContractDetails)
00041: class ContractDetailsAdmin(admin.ModelAdmin):
00042:     list_display = ['cdname', 'cdtrim', 'cdtrim_qty', 'cdrough', 'cdrough_qty', 'cduni
t_price', 'cdwork_price']
00043:     list_filter = ['cdwork_price']
00044:     search_fields = ['cdname', 'cdwork_price__name']
00045:
```

ctrctsapp\apps.py

```
00001: from django.apps import AppConfig
00002:
00003:
00004: class CtrctsappConfig(AppConfig):
00005:     default_auto_field = 'django.db.models.BigAutoField'
00006:     name = 'ctrctsapp'
00007:     verbose_name = 'Contracts Module'
```

ctrctsapp\management\commands\delete_expired_tokens.py

```
00001: from django.core.management.base import BaseCommand
00002: from datetime import timedelta, datetime
00003: from rest_framework_expiring_authtoken.models import ExpiringToken
00004:
00005: class Command(BaseCommand):
00006:     help = 'Delete expired tokens from the database'
00007:
00008:     def handle(self, *args, **kwargs):
00009:         expiration_time = timedelta(hours=8)  # Set the expiration time here
00010:         tokens_deleted = ExpiringToken.objects.filter(
00011:             created__lt=datetime.now() - expiration_time
00012:         ).delete()
00013:
00014:         self.stdout.write(self.style.SUCCESS(f'{tokens_deleted[0]} expired tokens dele
ted'))
```

ctrctsapp\management\mysqldump.py

```
00001: import os
00002: import subprocess
00003: import datetime
00004:
00005: # Configure your parameters
00006: user = 'root'
00007: password = 'Oliver.usa1017$'
00008: database_name = 'chalan_admin'
00009: backup_path = r'C:\Users\Division 16 - #33\Dropbox\Contract Paysheets\chalan_pro\backu
ps'
00010:
00011: # Create the backup directory if it does not exist
00012: if not os.path.exists(backup_path):
00013:     os.makedirs(backup_path)
00014:
00015: # Backup file name with date and time
00016: backup_file = os.path.join(backup_path, f"{database_name}_{datetime.datetime.now().str
ftime('%Y%m%d')}.sql")
00017: print(backup_file)
00018:
00019:
00020: # Prepare the command
00021: command = f'mysqldump -u {user} -p{password} {database_name} > "{backup_file}"'
00022:
00023: # Execute the command
00024: try:
00025:     # subprocess.run() is used to execute the command
00026:     result = subprocess.run(command, shell=True, check=True, text=True, capture_output
=True)
00027:     print("Backup completed successfully.")
00028: except subprocess.CalledProcessError as e:
00029:     print("Error executing the command:", e)
00030:     print("Standard output:", e.stdout)
00031:     print("Error output:", e.stderr)
```

ctrctsapp\migrations\0001_initial.py

```
00001: # Generated by Django 5.0.3 on 2024-04-17 01:29
00002:
00003: import django.db.models.deletion
00004: from django.conf import settings
00005: from django.db import migrations, models
00006:
00007:
00008: class Migration(migrations.Migration):
00009:
00010:     initial = True
00011:
00012:     dependencies = [
00013:         migrations.swappable_dependency(settings.AUTH_USER_MODEL),
00014:     ]
00015:
00016:     operations = [
00017:         migrations.CreateModel(
00018:             name='Builder',
00019:             fields=[
00020:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00021:                 ('name', models.CharField(max_length=255, null=True, verbose_name='Nam
e')),
00022:                 ('trim_amount', models.DecimalField(decimal_places=2, default=0, max_d
igits=10, verbose_name='Trim Price SqFt')),
00023:                 ('rough_amount', models.DecimalField(decimal_places=2, default=0, max_
digits=10, verbose_name='Rough Price SqFt')),
00024:                 ('travel_price_amount', models.DecimalField(decimal_places=2, default=
0, max_digits=10, verbose_name='Travel Amount')),
00025:             ],
00026:             options={
00027:                 'verbose_name': 'Builder',
00028:                 'verbose_name_plural': 'Builders',
00029:                 'ordering': ['name'],
00030:             },
00031:         ),
00032:         migrations.CreateModel(
00033:             name='Job',
00034:             fields=[
00035:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00036:                 ('name', models.CharField(max_length=255, verbose_name='Name')),
00037:                 ('builder', models.ForeignKey(null=True, on_delete=django.db.models.de
letion.CASCADE, related_name='jobs', to='ctrctsapp.builder')),
00038:             ],
00039:             options={
00040:                 'verbose_name': 'Community',
00041:                 'verbose_name_plural': 'Communities (Job)',
00042:                 'ordering': ['name'],
00043:             },
00044:         ),
00045:         migrations.CreateModel(
00046:             name='HouseModel',
00047:             fields=[
00048:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00049:                 ('name', models.CharField(max_length=255, verbose_name='Name')),
00050:                 ('jobs', models.ManyToManyField(related_name='house_models', to='ctrct
sapp.job')),
00051:             ],
```

ctrctsapp\migrations\0001_initial.py

```
00052:             options={
00053:                 'verbose_name': 'House Model',
00054:                 'verbose_name_plural': 'House Models',
00055:                 'ordering': ['name'],
00056:             },
00057:         ),
00058:         migrations.CreateModel(
00059:             name='Contract',
00060:             fields=[
00061:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00062:                 ('date_created', models.DateTimeField(auto_now_add=True)),
00063:                 ('last_updated', models.DateTimeField(auto_now=True)),
00064:                 ('type', models.CharField(choices=[('Rough', 'Rough'), ('Trim', 'Trim'
)], max_length=5)),
00065:                 ('lot', models.CharField(max_length=10, null=True, verbose_name='Lot')
),
00066:                 ('sqft', models.IntegerField(verbose_name='SqFt')),
00067:                 ('address', models.CharField(max_length=255, verbose_name='Address')),
00068:                 ('job_price', models.DecimalField(decimal_places=2, default=0, max_dig
its=10, verbose_name='Job Price')),
00069:                 ('travel_price', models.DecimalField(decimal_places=2, default=0, max_
digits=10, verbose_name='Travel Price')),
00070:                 ('total_options', models.DecimalField(decimal_places=2, default=0, max
_digits=10, null=True, verbose_name='Total Options')),
00071:                 ('total', models.DecimalField(decimal_places=2, default=0, max_digits=
10, null=True, verbose_name='Total')),
00072:                 ('comment', models.TextField(default='Required to finish at 100%', nul
l=True, verbose_name='Comment')),
00073:                 ('file', models.FileField(blank=True, default=None, null=True, upload_
to='contract', verbose_name='File')),
00074:                 ('builder', models.ForeignKey(on_delete=django.db.models.deletion.CASC
ADE, to='ctrctsapp.builder', verbose_name='Builder')),
00075:                 ('created_by', models.ForeignKey(null=True, on_delete=django.db.models
.deletion.SET_NULL, related_name='created_contracts', to=settings.AUTH_USER_MODEL, verbose_na
me='Created By')),
00076:                 ('house_model', models.ForeignKey(on_delete=django.db.models.deletion.
CASCADE, to='ctrctsapp.housemodel', verbose_name='House Model')),
00077:                 ('job', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
 to='ctrctsapp.job', verbose_name='Job')),
00078:             ],
00079:             options={
00080:                 'verbose_name': 'Contract',
00081:                 'verbose_name_plural': 'Contracts',
00082:                 'ordering': ['-date_created'],
00083:             },
00084:         ),
00085:         migrations.CreateModel(
00086:             name='WorkPrice',
00087:             fields=[
00088:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00089:                 ('name', models.CharField(max_length=255, verbose_name='Nombre')),
00090:                 ('trim', models.DecimalField(decimal_places=2, default=0, max_digits=1
0, verbose_name='Trim')),
00091:                 ('rough', models.DecimalField(decimal_places=2, default=0, max_digits=
10, verbose_name='Rough')),
00092:                 ('unit_price', models.CharField(max_length=255, verbose_name='Unit Pri
ce Type')),
00093:                 ('builders', models.ManyToManyField(related_name='work_prices', to='ct
```

ctrctsapp\migrations\0001_initial.py

```
rctsapp.builder', verbose_name='Builders')),
00094:             ],
00095:             options={
00096:                 'verbose_name': 'Work Price',
00097:                 'verbose_name_plural': 'Work Prices',
00098:                 'ordering': ['id'],
00099:             },
00100:         ),
00101:         migrations.CreateModel(
00102:             name='ContractDetails',
00103:             fields=[
00104:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00105:                 ('cdname', models.CharField(max_length=255, verbose_name='Nombre')),
00106:                 ('cdtrim', models.DecimalField(decimal_places=2, default=0, max_digits
=10, verbose_name='Trim Qty')),
00107:                 ('cdtrim_qty', models.DecimalField(decimal_places=2, default=0, max_di
gits=10, verbose_name='Trim')),
00108:                 ('cdrough', models.DecimalField(decimal_places=2, default=0, max_digit
s=10, verbose_name='Rough')),
00109:                 ('cdrough_qty', models.DecimalField(decimal_places=2, default=0, max_d
igits=10, verbose_name='Rough Qty')),
00110:                 ('cdunit_price', models.CharField(max_length=255, verbose_name='Unit P
rice Type')),
00111:                 ('contract_details', models.ForeignKey(null=True, on_delete=django.db.
models.deletion.CASCADE, related_name='contract_details', to='ctrctsapp.contract', verbose_na
me='Work Price Details')),
00112:                 ('cdwork_price', models.ForeignKey(null=True, on_delete=django.db.mode
ls.deletion.CASCADE, related_name='priceDetails', to='ctrctsapp.workprice', verbose_name='Pri
ce Details')),
00113:             ],
00114:             options={
00115:                 'verbose_name': 'Contract Detail',
00116:                 'verbose_name_plural': 'Contract Details',
00117:                 'ordering': ['-id'],
00118:             },
00119:         ),
00120:     ]
```

ctrctsapp\migrations\0002_alter_contract_options_alter_contract_type.py

```
00001: # Generated by Django 5.0.3 on 2024-05-29 21:29
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('ctrctsapp', '0001_initial'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterModelOptions(
00014:             name='contract',
00015:             options={'ordering': ['-date_created'], 'verbose_name': 'Contract', 'verbo
se_name_plural': 'Contracts'},
00016:         ),
00017:         migrations.AlterField(
00018:             model_name='contract',
00019:             name='type',
00020:             field=models.CharField(choices=[('Rough', 'Rough'), ('Trim', 'Trim')], max
_length=5),
00021:         ),
00022:     ]
```

ctrctsapp\migrations\0003_contractdetails_cdrough_qty_and_more.py

```
00001: # Generated by Django 5.0.3 on 2024-05-29 21:29
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('ctrctsapp', '0002_alter_contract_options_alter_contract_type'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AddField(
00014:             model_name='contractdetails',
00015:             name='cdrough_qty',
00016:             field=models.DecimalField(decimal_places=2, default=0, max_digits=10, verb
ose_name='Rough Qty'),
00017:         ),
00018:         migrations.AddField(
00019:             model_name='contractdetails',
00020:             name='cdtrim_qty',
00021:             field=models.DecimalField(decimal_places=2, default=0, max_digits=10, verb
ose_name='Trim'),
00022:         ),
00023:         migrations.AlterField(
00024:             model_name='contractdetails',
00025:             name='cdtrim',
00026:             field=models.DecimalField(decimal_places=2, default=0, max_digits=10, verb
ose_name='Trim Qty'),
00027:         ),
00028:     ]
```

ctrctsapp\migrations\0004_builder_housemodel_workprice_builder_and_more.py

```
00001: # Generated by Django 5.0.3 on 2024-06-08 17:57
00002:
00003: import django.db.models.deletion
00004: from django.db import migrations, models
00005:
00006:
00007: class Migration(migrations.Migration):
00008:
00009:     dependencies = [
00010:         ('ctrctsapp', '0003_contractdetails_cdrough_qty_and_more'),
00011:     ]
00012:
00013:     operations = [
00014:         migrations.CreateModel(
00015:             name='Builder',
00016:             fields=[
00017:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00018:                 ('name', models.CharField(max_length=255, verbose_name='Nombre')),
00019:             ],
00020:             options={
00021:                 'verbose_name': 'Builder',
00022:                 'verbose_name_plural': 'Builders',
00023:                 'ordering': ['name'],
00024:             },
00025:         ),
00026:         migrations.CreateModel(
00027:             name='HouseModel',
00028:             fields=[
00029:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00030:                 ('name', models.CharField(max_length=255, verbose_name='Nombre')),
00031:             ],
00032:             options={
00033:                 'verbose_name': 'House Model',
00034:                 'verbose_name_plural': 'House Models',
00035:                 'ordering': ['name'],
00036:             },
00037:         ),
00038:         migrations.AddField(
00039:             model_name='workprice',
00040:             name='builder',
00041:             field=models.ForeignKey(null=True, on_delete=django.db.models.deletion.CAS
CADE, related_name='work_prices', to='ctrctsapp.builder', verbose_name='Builder'),
00042:         ),
00043:         migrations.AlterField(
00044:             model_name='contract',
00045:             name='builder',
00046:             field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='c
trctsapp.builder', verbose_name='Builder'),
00047:         ),
00048:         migrations.AlterField(
00049:             model_name='contract',
00050:             name='house_model',
00051:             field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='c
trctsapp.housemodel', verbose_name='House Model'),
00052:         ),
00053:         migrations.CreateModel(
00054:             name='Job',
00055:             fields=[
```

ctrctsapp\migrations\0004_builder_housemodel_workprice_builder_and_more.py

```
00056:                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serial
ize=False, verbose_name='ID')),
00057:                 ('name', models.CharField(max_length=255, verbose_name='Nombre')),
00058:                 ('builder', models.ForeignKey(null=True, on_delete=django.db.models.de
letion.CASCADE, related_name='jobs', to='ctrctsapp.builder', verbose_name='Builder')),
00059:             ],
00060:             options={
00061:                 'verbose_name': 'Job',
00062:                 'verbose_name_plural': 'Jobs',
00063:                 'ordering': ['name'],
00064:             },
00065:         ),
00066:         migrations.AddField(
00067:             model_name='housemodel',
00068:             name='job',
00069:             field=models.ForeignKey(null=True, on_delete=django.db.models.deletion.CAS
CADE, related_name='house_models', to='ctrctsapp.job', verbose_name='Job'),
00070:         ),
00071:         migrations.AlterField(
00072:             model_name='contract',
00073:             name='job',
00074:             field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='c
trctsapp.job', verbose_name='Job'),
00075:         ),
00076:     ]
```

ctrctsapp\migrations\0005_remove_housemodel_job_housemodel_jobs_job_address_and_more.py

```
00001: # Generated by Django 5.0.3 on 2024-06-08 20:58
00002:
00003: import django.db.models.deletion
00004: from django.db import migrations, models
00005:
00006:
00007: class Migration(migrations.Migration):
00008:
00009:     dependencies = [
00010:         ('ctrctsapp', '0004_builder_housemodel_workprice_builder_and_more'),
00011:     ]
00012:
00013:     operations = [
00014:         migrations.RemoveField(
00015:             model_name='housemodel',
00016:             name='job',
00017:         ),
00018:         migrations.AddField(
00019:             model_name='housemodel',
00020:             name='jobs',
00021:             field=models.ManyToManyField(related_name='house_models', to='ctrctsapp.jo
b'),
00022:         ),
00023:         migrations.AddField(
00024:             model_name='job',
00025:             name='address',
00026:             field=models.CharField(default=1, max_length=255, verbose_name='Address'),
00027:             preserve_default=False,
00028:         ),
00029:         migrations.AlterField(
00030:             model_name='builder',
00031:             name='name',
00032:             field=models.CharField(max_length=255, null=True, verbose_name='Name'),
00033:         ),
00034:         migrations.AlterField(
00035:             model_name='housemodel',
00036:             name='name',
00037:             field=models.CharField(max_length=255, verbose_name='Name'),
00038:         ),
00039:         migrations.AlterField(
00040:             model_name='job',
00041:             name='builder',
00042:             field=models.ForeignKey(null=True, on_delete=django.db.models.deletion.CAS
CADE, related_name='jobs', to='ctrctsapp.builder'),
00043:         ),
00044:         migrations.AlterField(
00045:             model_name='job',
00046:             name='name',
00047:             field=models.CharField(max_length=255, verbose_name='Name'),
00048:         ),
00049:     ]
```

ctrctsapp\migrations\0006_alter_job_options_remove_workprice_builder_and_more.py

```
00001: # Generated by Django 5.0.3 on 2024-06-09 16:36
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('ctrctsapp', '0005_remove_housemodel_job_housemodel_jobs_job_address_and_more
'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterModelOptions(
00014:             name='job',
00015:             options={'ordering': ['name'], 'verbose_name': 'Community', 'verbose_name_
plural': 'Communities'},
00016:         ),
00017:         migrations.RemoveField(
00018:             model_name='workprice',
00019:             name='builder',
00020:         ),
00021:         migrations.AddField(
00022:             model_name='workprice',
00023:             name='builders',
00024:             field=models.ManyToManyField(related_name='work_prices', to='ctrctsapp.bui
lder', verbose_name='Builders'),
00025:         ),
00026:     ]
```

ctrctsapp\migrations\0007_remove_job_address.py

```
00001: # Generated by Django 5.0.3 on 2024-06-09 17:44
00002:
00003: from django.db import migrations
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('ctrctsapp', '0006_alter_job_options_remove_workprice_builder_and_more'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.RemoveField(
00014:             model_name='job',
00015:             name='address',
00016:         ),
00017:     ]
```

ctrctsapp\migrations\0008_builder_rough_amount_builder_trim_amount.py

```
00001: # Generated by Django 5.0.3 on 2024-06-19 00:44
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('ctrctsapp', '0007_remove_job_address'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AddField(
00014:             model_name='builder',
00015:             name='rough_amount',
00016:             field=models.DecimalField(decimal_places=2, default=0, max_digits=10, verb
ose_name='Rough Amount'),
00017:         ),
00018:         migrations.AddField(
00019:             model_name='builder',
00020:             name='trim_amount',
00021:             field=models.DecimalField(decimal_places=2, default=0, max_digits=10, verb
ose_name='Trim Amount'),
00022:         ),
00023:     ]
```

ctrctsapp\migrations\0009_builder_travel_price_amount.py

```
00001: # Generated by Django 5.0.3 on 2024-06-19 23:51
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('ctrctsapp', '0008_builder_rough_amount_builder_trim_amount'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AddField(
00014:             model_name='builder',
00015:             name='travel_price_amount',
00016:             field=models.DecimalField(decimal_places=2, default=0, max_digits=10, verb
ose_name='Rough Amount'),
00017:         ),
00018:     ]
```

ctrctsapp\migrations\0010_alter_job_options_alter_workprice_options_and_more.py

```
00001: # Generated by Django 5.0.3 on 2024-08-04 20:43
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('ctrctsapp', '0009_builder_travel_price_amount'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AlterModelOptions(
00014:             name='job',
00015:             options={'ordering': ['name'], 'verbose_name': 'Community', 'verbose_name_
plural': 'Communities (Job)'},
00016:         ),
00017:         migrations.AlterModelOptions(
00018:             name='workprice',
00019:             options={'ordering': ['id'], 'verbose_name': 'Work Price', 'verbose_name_p
lural': 'Work Prices'},
00020:         ),
00021:         migrations.AlterField(
00022:             model_name='builder',
00023:             name='rough_amount',
00024:             field=models.DecimalField(decimal_places=2, default=0, max_digits=10, verb
ose_name='Rough Price SqFt'),
00025:         ),
00026:         migrations.AlterField(
00027:             model_name='builder',
00028:             name='travel_price_amount',
00029:             field=models.DecimalField(decimal_places=2, default=0, max_digits=10, verb
ose_name='Travel Amount'),
00030:         ),
00031:         migrations.AlterField(
00032:             model_name='builder',
00033:             name='trim_amount',
00034:             field=models.DecimalField(decimal_places=2, default=0, max_digits=10, verb
ose_name='Trim Price SqFt'),
00035:         ),
00036:         migrations.AlterField(
00037:             model_name='contract',
00038:             name='lot',
00039:             field=models.CharField(max_length=10, null=True, verbose_name='Lot'),
00040:         ),
00041:     ]
```

ctrctsapp\migrations\0011_job_address_job_latitude_job_longitude.py

```
00001: # Generated by Django 5.0.3 on 2025-01-10 02:56
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('ctrctsapp', '0010_alter_job_options_alter_workprice_options_and_more'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AddField(
00014:             model_name='job',
00015:             name='address',
00016:             field=models.CharField(blank=True, max_length=255, null=True, verbose_name
='Address'),
00017:         ),
00018:         migrations.AddField(
00019:             model_name='job',
00020:             name='latitude',
00021:             field=models.DecimalField(blank=True, decimal_places=6, max_digits=9, null
=True, verbose_name='Latitude'),
00022:         ),
00023:         migrations.AddField(
00024:             model_name='job',
00025:             name='longitude',
00026:             field=models.DecimalField(blank=True, decimal_places=6, max_digits=9, null
=True, verbose_name='Longitude'),
00027:         ),
00028:     ]
```

ctrctsapp\migrations\0012_contract_doc_type_contract_needs_reprint.py

```
00001: # Generated by Django 5.0.3 on 2025-05-26 05:12
00002:
00003: from django.db import migrations, models
00004:
00005:
00006: class Migration(migrations.Migration):
00007:
00008:     dependencies = [
00009:         ('ctrctsapp', '0011_job_address_job_latitude_job_longitude'),
00010:     ]
00011:
00012:     operations = [
00013:         migrations.AddField(
00014:             model_name='contract',
00015:             name='doc_type',
00016:             field=models.CharField(choices=[('Contract', 'Contract'), ('Bid', 'Bid')],
 default='Contract', max_length=10),
00017:         ),
00018:         migrations.AddField(
00019:             model_name='contract',
00020:             name='needs_reprint',
00021:             field=models.BooleanField(default=False),
00022:         ),
00023:     ]
```

ctrctsapp\migrations__init__.py

```
00001:
```

ctrctsapp\models.py

```
00001: from django.db import models
00002: from django.contrib.auth.models import User
00003: from django.utils import timezone
00004: from ctrctsapp.utils import geocode_address
00005:
00006: # Modelo de Builder.
00007: class Builder(models.Model):
00008:     name = models.CharField(max_length=255, null=True, verbose_name='Name')
00009:     trim_amount = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='T
rim Price SqFt', default=0)
00010:     rough_amount = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='
Rough Price SqFt', default=0)
00011:     travel_price_amount = models.DecimalField(max_digits=10, decimal_places=2, verbose
_name='Travel Amount', default=0)
00012:     # Otros campos relevantes del builder
00013:
00014:     class Meta:
00015:         verbose_name = 'Builder'
00016:         verbose_name_plural = 'Builders'
00017:         ordering = ['name']
00018:
00019:     def __str__(self):
00020:         return self.name
00021:
00022: # Modelo de Job.
00023: class Job(models.Model):
00024:     name = models.CharField(max_length=255, verbose_name='Name')
00025:     builder = models.ForeignKey(Builder, null=True, on_delete=models.CASCADE, related_
name='jobs')
00026:     address = models.CharField(max_length=255, null=True, blank=True, verbose_name='Ad
dress')  # Nueva dirección
00027:     latitude = models.DecimalField(max_digits=9, decimal_places=6, null=True, blank=Tr
ue, verbose_name='Latitude')
00028:     longitude = models.DecimalField(max_digits=9, decimal_places=6, null=True, blank=T
rue, verbose_name='Longitude')
00029:     # Otros campos relevantes del job
00030:
00031:
00032:     def save(self, *args, **kwargs):
00033:         # Si hay dirección y no hay coordenadas, calcularlas
00034:         if self.address and (not self.latitude or not self.longitude):
00035:             self.latitude, self.longitude = geocode_address(self.address)
00036:         super().save(*args, **kwargs)
00037:
00038:     class Meta:
00039:         verbose_name = 'Community'
00040:         verbose_name_plural = 'Communities (Job)'
00041:         ordering = ['name']
00042:
00043:     def __str__(self):
00044:         return self.name
00045:
00046: # Modelo de House Model.
00047: class HouseModel(models.Model):
00048:     name = models.CharField(max_length=255, verbose_name='Name')
00049:     jobs = models.ManyToManyField(Job, related_name='house_models')
00050:     # Otros campos relevantes del house model
00051:
00052:     class Meta:
00053:         verbose_name = 'House Model'
```

ctrctsapp\models.py

```
00054:         verbose_name_plural = 'House Models'
00055:         ordering = ['name']
00056:
00057:     def __str__(self):
00058:         return self.name
00059:
00060: # Modelo de Work Price.
00061: class WorkPrice(models.Model):
00062:     name = models.CharField(max_length=255, verbose_name='Nombre')
00063:     trim = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Trim', d
efault=0)
00064:     rough = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Rough',
 default=0)
00065:     unit_price = models.CharField(max_length=255, verbose_name='Unit Price Type')
00066:     builders = models.ManyToManyField(Builder, related_name='work_prices', verbose_nam
e='Builders')
00067:
00068:     class Meta:
00069:         verbose_name = 'Work Price'
00070:         verbose_name_plural = 'Work Prices'
00071:         ordering = ['id']
00072:
00073:     def __str__(self):
00074:         return self.name
00075:
00076: # Modelo de Work this.contract.
00077: class Contract(models.Model):
00078:     TYPE_CHOICES = [
00079:         ('Rough', 'Rough'),
00080:         ('Trim', 'Trim'),
00081:     ]
00082:
00083:     DOC_TYPE_CHOICES = [
00084:     ('Contract', 'Contract'),
00085:     ('Bid', 'Bid'),
00086:     ]
00087:
00088:     date_created = models.DateTimeField(auto_now_add=timezone.now)
00089:     last_updated = models.DateTimeField(auto_now=True)
00090:     type = models.CharField(max_length=5, choices=TYPE_CHOICES)
00091:     doc_type = models.CharField(max_length=10, choices=DOC_TYPE_CHOICES, default='Cont
ract')
00092:     builder = models.ForeignKey(Builder, on_delete=models.CASCADE, verbose_name='Build
er')
00093:     house_model = models.ForeignKey(HouseModel, on_delete=models.CASCADE, verbose_name
='House Model')
00094:     job = models.ForeignKey(Job, on_delete=models.CASCADE, verbose_name='Job')
00095:     lot = models.CharField(null=True, max_length=10, verbose_name='Lot')
00096:     sqft = models.IntegerField(verbose_name='SqFt')
00097:     address = models.CharField(max_length=255, verbose_name='Address')
00098:     job_price = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Job
 Price', default=0)
00099:     travel_price = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='
Travel Price', default=0)
00100:     total_options = models.DecimalField(null=True, max_digits=10, decimal_places=2, ve
rbose_name='Total Options', default=0)
00101:     total = models.DecimalField(null=True, max_digits=10, decimal_places=2, verbose_na
me='Total', default=0)
00102:     comment = models.TextField(null=True, verbose_name='Comment', default='Required to
 finish at 100%')
```

ctrctsapp\models.py

```
00103:     file = models.FileField(null=True, upload_to='contract', default=None, blank=True,
 verbose_name='File')
00104:     created_by = models.ForeignKey(User, on_delete=models.SET_NULL, null=True, related
_name='created_contracts', verbose_name='Created By')
00105:     needs_reprint = models.BooleanField(default=False)
00106:
00107:     class Meta:
00108:         verbose_name = 'Contract'
00109:         verbose_name_plural = 'Contracts'
00110:         ordering = ['-date_created']
00111:
00112:     def __str__(self):
00113:         return f"Contract {self.id} - {self.created_by}"
00114:
00115: # Modelo de Work Details this.contract.
00116: class ContractDetails(models.Model):
00117:     cdname = models.CharField(max_length=255, verbose_name='Nombre')
00118:     cdtrim = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Trim Q
ty', default=0)
00119:     cdtrim_qty = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Tr
im', default=0)
00120:     cdrough = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='Rough
', default=0)
00121:     cdrough_qty = models.DecimalField(max_digits=10, decimal_places=2, verbose_name='R
ough Qty', default=0)
00122:     cdunit_price = models.CharField(max_length=255, verbose_name='Unit Price Type')
00123:     cdwork_price = models.ForeignKey(WorkPrice, on_delete=models.CASCADE, related_name
='priceDetails', verbose_name='Price Details', null=True)
00124:     contract_details = models.ForeignKey(Contract, on_delete=models.CASCADE, related_n
ame='contract_details', verbose_name='Work Price Details', null=True)
00125:
00126:     class Meta:
00127:         verbose_name = 'Contract Detail'
00128:         verbose_name_plural = 'Contract Details'
00129:         ordering = ['-id']
00130:
00131:     def __str__(self):
00132:         return self.cdname
```

ctrctsapp\serializers.py

```
00001: from django.core.exceptions import ObjectDoesNotExist
00002: from django.db import transaction
00003: from rest_framework import serializers, viewsets
00004: from .models import WorkPrice, Contract, ContractDetails, Builder, Job, HouseModel
00005: from crewsapp.models import Crew
00006:
00007: import logging
00008:
00009: logger = logging.getLogger(__name__)
00010:
00011: # Serializer for the Builder model
00012: # Serializador para el modelo Builder
00013: class BuilderSerializer(serializers.ModelSerializer):
00014:     class Meta:
00015:         model = Builder
00016:         fields = ['id', 'name', 'trim_amount', 'rough_amount', 'travel_price_amount']
00017:
00018: # Mini serializer de lectura (opcional, para debug/UX)
00019: class CrewMiniSerializer(serializers.ModelSerializer):
00020:     class Meta:
00021:         model = Crew
00022:         fields = ("id", "name")
00023:
00024: class JobSerializer(serializers.ModelSerializer):
00025:     builder = serializers.PrimaryKeyRelatedField(queryset=Builder.objects.all())
00026:
00027:     # Acepta escribir crews como lista de IDs
00028:     crews = serializers.PrimaryKeyRelatedField(
00029:         queryset=Crew.objects.all(),
00030:         many=True,
00031:         required=False,
00032:         write_only=True  # solo escritura; si quieres verlo en responses, quítalo
00033:     )
00034:
00035:     # Detalle de lectura opcional (no estorba)
00036:     crews_detail = CrewMiniSerializer(source="crews", many=True, read_only=True)
00037:
00038:     # (opcional) compatibilidad con tu campo anterior de solo lectura
00039:     crew_leaders = serializers.SerializerMethodField(read_only=True)
00040:
00041:     class Meta:
00042:         model = Job
00043:         fields = [
00044:             "id", "name", "builder", "address", "latitude", "longitude",
00045:             "crews",          # << === escribe M2M
00046:             "crews_detail",   # << === lee M2M (id+name)
00047:             "crew_leaders",   # << === legacy read-only
00048:         ]
00049:
00050:     def get_crew_leaders(self, obj):
00051:         return list(obj.crews.values_list("name", flat=True))
00052:
00053:     # Blindaje: setear la M2M explícitamente en create/update
00054:     def create(self, validated_data):
00055:         crews = validated_data.pop("crews", [])
00056:         job = super().create(validated_data)
00057:         if crews:
00058:             job.crews.set(crews)
00059:         return job
00060:
```

ctrctsapp\serializers.py

```
00061:     def update(self, instance, validated_data):
00062:         crews = validated_data.pop("crews", None)
00063:         job = super().update(instance, validated_data)
00064:         if crews is not None:
00065:             job.crews.set(crews)
00066:         return job
00067:
00068: class JobViewSet(viewsets.ModelViewSet):
00069:     queryset = Job.objects.all().select_related('builder').prefetch_related('crews')
# performance
00070:     serializer_class = JobSerializer
00071:
00072:
00073: # Serializer for the HouseModel model, including related jobs
00074: class HouseModelSerializer(serializers.ModelSerializer):
00075:     jobs = serializers.PrimaryKeyRelatedField(queryset=Job.objects.all(), many=True)
00076:
00077:     class Meta:
00078:         model = HouseModel
00079:         fields = ['id', 'name', 'jobs']
00080:
00081:     def create(self, validated_data):
00082:         # Extract jobs from validated data
00083:         jobs = validated_data.pop('jobs', [])
00084:         # Create the HouseModel instance
00085:         house_model = HouseModel.objects.create(**validated_data)
00086:         # Add the jobs to the ManyToMany field
00087:         house_model.jobs.set(jobs)
00088:         return house_model
00089:
00090:     def update(self, instance, validated_data):
00091:         # Extract jobs from validated data
00092:         jobs = validated_data.pop('jobs', [])
00093:         # Update the HouseModel instance
00094:         instance.name = validated_data.get('name', instance.name)
00095:         instance.save()
00096:         # Update the ManyToMany field
00097:         instance.jobs.set(jobs)
00098:         return instance
00099:
00100: # Serializer for the ContractDetails model
00101: # Serializador para el modelo ContractDetails
00102: class ContractDetailsSerializer(serializers.ModelSerializer):
00103:     class Meta:
00104:         model = ContractDetails
00105:         fields = ['id', 'cdname', 'cdtrim', 'cdrough', 'cdunit_price', 'cdwork_price',
 'contract_details', 'cdtrim_qty', 'cdrough_qty']
00106:
00107: # Serializer for the WorkPrice model
00108: # Serializador para el modelo WorkPrice
00109: class WorkPriceSerializer(serializers.ModelSerializer):
00110:     builders = serializers.PrimaryKeyRelatedField(many=True, queryset=Builder.objects.
all())
00111:
00112:     class Meta:
00113:         model = WorkPrice
00114:         fields = ['id', 'name', 'trim', 'rough', 'unit_price', 'builders']
00115:
00116:     # Create a new WorkPrice instance and set its builders
00117:     # Crear una nueva instancia de WorkPrice y asignar sus builders
```

ctrctsapp\serializers.py

```
00118:     def create(self, validated_data):
00119:         builders_data = validated_data.pop('builders')
00120:         work_price = WorkPrice.objects.create(**validated_data)
00121:         work_price.builders.set(builders_data)
00122:         return work_price
00123:
00124:     # Update an existing WorkPrice instance and set its builders
00125:     # Actualizar una instancia existente de WorkPrice y asignar sus builders
00126:     def update(self, instance, validated_data):
00127:         builders_data = validated_data.pop('builders', None)
00128:
00129:         for attr, value in validated_data.items():
00130:             setattr(instance, attr, value)
00131:         instance.save()
00132:
00133:         if builders_data is not None:
00134:             instance.builders.set(builders_data)
00135:         return instance
00136:
00137: # Serializer for the Contract model
00138: # Serializador para el modelo Contract
00139: class ContractSerializer(serializers.ModelSerializer):
00140:     contract_details = ContractDetailsSerializer(many=True)
00141:     house_model = HouseModelSerializer(read_only=True)
00142:     builder = BuilderSerializer(read_only=True)
00143:     job = JobSerializer(read_only=True)
00144:     house_model_id = serializers.PrimaryKeyRelatedField(queryset=HouseModel.objects.al
l(), source='house_model', write_only=True)
00145:     builder_id = serializers.PrimaryKeyRelatedField(queryset=Builder.objects.all(), so
urce='builder', write_only=True)
00146:     job_id = serializers.PrimaryKeyRelatedField(queryset=Job.objects.all(), source='jo
b', write_only=True)
00147:
00148:     class Meta:
00149:         model = Contract
00150:         fields = [
00151:             'id', 'created_by', 'date_created', 'last_updated', 'type', 'builder', 'bu
ilder_id',
00152:             'job', 'job_id', 'house_model', 'house_model_id', 'lot', 'sqft', 'address'
, 'job_price', 'travel_price',
00153:             'total_options', 'total', 'comment', 'file', 'contract_details', 'needs_re
print', 'doc_type',
00154:         ]
00155:         read_only_fields = ['id', 'date_created', 'last_updated']
00156:
00157:     # Create a new Contract instance with validated data
00158:     # Crear una nueva instancia de Contract con datos validados
00159:     def create(self, validated_data):
00160:         contract_details_data = validated_data.pop('contract_details')
00161:         builder = validated_data.pop('builder')
00162:         job = validated_data.pop('job')
00163:         house_model = validated_data.pop('house_model')
00164:
00165:         validated_data['travel_price'] = builder.travel_price_amount  # Usa el travel_
price_amount del builder
00166:
00167:         with transaction.atomic():
00168:             # Create the contract with validated data
00169:             # Crear el contrato con los datos validados
00170:             contract = Contract.objects.create(
```

ctrctsapp\serializers.py

```
00171:                 **validated_data,
00172:                 builder=builder,
00173:                 job=job,
00174:                 house_model=house_model
00175:             )
00176:
00177:             for detail_data in contract_details_data:
00178:                 detail_data['contract_details_id'] = contract.id
00179:
00180:                 ContractDetails.objects.create(**detail_data)
00181:
00182:         return contract
00183:
00184:     # Update an existing Contract instance with validated data
00185:     # Actualizar una instancia existente de Contract con datos validados
00186:     def update(self, instance, validated_data):
00187:         # Extract nested data
00188:         # Extraer datos anidados
00189:         contract_details_data = validated_data.pop('contract_details', None)
00190:         builder_id = validated_data.pop('builder_id', None)
00191:         job_id = validated_data.pop('job_id', None)
00192:         house_model_id = validated_data.pop('house_model_id', None)
00193:
00194:         with transaction.atomic():
00195:             # Update foreign key relationships if they exist
00196:             # Actualizar relaciones de clave foránea si existen
00197:             if builder_id:
00198:                 instance.builder = Builder.objects.get(id=builder_id)
00199:                 instance.travel_price = instance.builder.travel_price_amount  # Usa el
 travel_price_amount del builder
00200:
00201:             if job_id:
00202:                 instance.job = Job.objects.get(id=job_id)
00203:
00204:             if house_model_id:
00205:                 instance.house_model = HouseModel.objects.get(id=house_model_id)
00206:
00207:             # Update other contract fields
00208:             # Actualizar otros campos del contrato
00209:             for attr, value in validated_data.items():
00210:                 setattr(instance, attr, value)
00211:             instance.save()
00212:
00213:             # Handle contract details
00214:             # Manejar los detalles del contrato
00215:             if contract_details_data is not None:
00216:                 # Delete existing details
00217:                 # Eliminar detalles existentes
00218:                 ContractDetails.objects.filter(contract_details=instance).delete()
00219:                 # Create new details
00220:                 # Crear nuevos detalles
00221:                 for detail_data in contract_details_data:
00222:                     # Ensure the 'contract_details' field is not passed twice
00223:                     # Asegúrate de no pasar el campo 'contract_details' dos veces
00224:                     ContractDetails.objects.create(contract_details=instance, **detail
_data)
00225:
00226:         return instance
00227:
00228: # Serializer for the Job model filtered by builder
```

ctrctsapp\serializers.py

```
00229: # Serializador para el modelo Job filtrado por builder
00230: class JobFilteredByBuilderSerializer(serializers.ModelSerializer):
00231:     class Meta:
00232:         model = Job
00233:         fields = ['id', 'name', 'builder', 'address', 'latitude', 'longitude']
00234:
00235: # Serializer for the HouseModel model filtered by job
00236: # Serializador para el modelo HouseModel filtrado por job
00237: class HouseModelFilteredByJobSerializer(serializers.ModelSerializer):
00238:     class Meta:
00239:         model = HouseModel
00240:         fields = ['id', 'name', 'jobs']
00241:
00242:
00243: # Serializers for list options
00244: class BuilderListSerializer(serializers.ModelSerializer):
00245:     class Meta:
00246:         model = Builder
00247:         fields = ['id', 'name']
00248:
00249:
00250: class JobListSerializer(serializers.ModelSerializer):
00251:     class Meta:
00252:         model = Job
00253:         fields = ['id', 'name', 'address', 'builder']
00254:
00255:
00256: class HouseListSerializer(serializers.ModelSerializer):
00257:     class Meta:
00258:         model = HouseModel
00259:         fields = ['id', 'name', 'jobs']
00260:
00261: class ContractListSerializer(serializers.ModelSerializer):
00262:     builder = BuilderSerializer()
00263:     job = JobSerializer()
00264:     house_model = HouseModelSerializer()
00265:
00266:     class Meta:
00267:         model = Contract
00268:         fields = ['id', 'type', 'builder', 'job', 'house_model', 'lot', 'address', 'sq
ft', 'job_price', 'total_options', 'total', 'needs_reprint', 'doc_type', 'date_created']
```

ctrctsapp\static

00001:

ctrctsapp\templates\contract_pdf.html

```
00001: <!DOCTYPE html>
00002: <html lang="en">
00003: <head>
00004:     <meta charset="UTF-8">
00005:     <meta name="viewport" content="width=device-width, initial-scale=1">
00006:     <title>Contract {{ contract.pk }}</title>
00007:
00008:     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet"
00009:           integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+
ALEwIH" crossorigin="anonymous">
00010:     <style>
00011:         @page {
00012:             size: "A4";
00013:             margin: 1.0cm 1.0cm 1.0cm 1.0cm;
00014:         }
00015:
00016:         .logo {
00017:             width: 170px;
00018:             margin-bottom: 10px;
00019:             float: left;
00020:         }
00021:
00022:         body {
00023:             width: 100% !important;
00024:             height: 100%;
00025:             background: #fff;
00026:             color: black;
00027:             font-size: 13px;
00028:             font-family: 'Roboto', sans-serif;
00029:             -webkit-font-smoothing: antialiased;
00030:             -webkit-text-size-adjust: none;
00031:         }
00032:         .text-justify {
00033:             text-align: justify;
00034:         }
00035:
00036:
00037:     </style>
00038: </head>
00039: <body>
00040: <div class="container">
00041:
00042:     {% if logo_url %}
00043:         <img src="{{ logo_url }}" class="logo" />
00044:     {% endif %}
00045:
00046:     <!-- Title -->
00047:     <div class="row">
00048:         <div class="col-12 text-center mb-4">
00049:             <h2 class="fw-bold">CONTRACT PAY SHEET - {{ contract.type | upper }}</h2>
00050:         </div>
00051:     </div>
00052:
00053:     <!-- Contract Details -->
00054:     <div class="row mb-3">
00055:         <!-- Left Column -->
00056:         <div class="col-6">
00057:             <strong>BUILDERS:</strong> {{ contract.builder.name }} <br/>
00058:             <strong>JOB:</strong> {{ contract.job.name }} <br/>
```

ctrctsapp\templates\contract_pdf.html

```
00059:             <strong>HOUSE MODEL:</strong> {{ contract.house_model.name }} <br/>
00060:             <strong>ADDRESS:</strong> {{ contract.address }} <br/>
00061:             <strong>LOT:</strong> {{ contract.lot }} <br/>
00062:             <strong>SQFT:</strong> {{ contract.sqft }} <br/>
00063:         </div>
00064:         <!-- Right Column -->
00065:         <div class="col-6">
00066:             <strong>DATE CREATED:</strong> {{ contract.date_created |date:"F j, Y" }}
<br/>
00067:             <strong>CONTRACT ID:</strong> {{ contract.id }} <br/>
00068:             {% if contract.comment %}
00069:                 <div class="p-1 bg-info bg-opacity-10 border border-info">
00070:                     <strong>ADDITIONAL INFO:</strong>
00071:                     <p class="mb-0">{{ contract.comment }}</p>
00072:                 </div>
00073:             {% endif %}
00074:             <strong>JOB PRICE: </strong> ${{ contract.job_price }} <br/>
00075:             <strong>TRAVEL PRICE: </strong>$ {{ contract.travel_price }} <br/>
00076:         </div>
00077:     </div>
00078:
00079:     <!-- Options -->
00080:     <div class="row mb-4">
00081:         <div class="col-12 text-center fs-4">
00082:             OPTIONS {{ contract.type | upper }}
00083:         </div>
00084:     </div>
00085:
00086:     <!-- Options Table (Two Columns) -->
00087:     <div class="row mb-4 justify-content-evenly" style="font-size: 10px">
00088:         <!-- Left Column -->
00089:         <div class="col-5">
00090:             <table class="table table-sm">
00091:                 <thead>
00092:                 <tr>
00093:                     <th>QTY</th>
00094:                     <th class="text-start">OPTION</th>
00095:                     <th class="text-end">AMOUNT</th>
00096:                 </tr>
00097:                 </thead>
00098:                 <tbody>
00099:                 {% for detail in left_details %}
00100:                     {% if contract.type == "Trim" and detail.cdtrim > 0 %}
00101:                         <tr>
00102:                             <td>{{ detail.cdtrim_qty }}</td>
00103:                             <td class="text-start">{{ detail.cdname }}</td>
00104:                             <td class="text-end">$ {{ detail.cdtrim }}</td>
00105:                         </tr>
00106:                     {% elif contract.type == "Rough" and detail.cdrough > 0 %}
00107:                         <tr>
00108:                             <td>{{ detail.cdrough_qty }}</td>
00109:                             <td class="text-start">{{ detail.cdname }}</td>
00110:                             <td class="text-end">$ {{ detail.cdrough }}</td>
00111:                         </tr>
00112:                     {% endif %}
00113:                 {% endfor %}
00114:                 </tbody>
00115:             </table>
00116:         </div>
00117:
```

ctrctsapp\templates\contract_pdf.html

```
00118:         <!-- Right Column -->
00119:         <div class="col-5">
00120:             <table class="table table-sm">
00121:                 <thead>
00122:                 <tr>
00123:                     <th>QTY</th>
00124:                     <th class="text-start">OPTION</th>
00125:                     <th class="text-end">AMOUNT</th>
00126:                 </tr>
00127:                 </thead>
00128:                 <tbody>
00129:                 {% for detail in right_details %}
00130:                     {% if contract.type == "Trim" and detail.cdtrim > 0 %}
00131:                         <tr>
00132:                             <td>{{ detail.cdtrim_qty }}</td>
00133:                             <td class="text-start">{{ detail.cdname }}</td>
00134:                             <td class="text-end">$ {{ detail.cdtrim }}</td>
00135:                         </tr>
00136:                     {% elif contract.type == "Rough" and detail.cdrough > 0 %}
00137:                         <tr>
00138:                             <td>{{ detail.cdrough_qty }}</td>
00139:                             <td class="text-start">{{ detail.cdname }}</td>
00140:                             <td class="text-end">$ {{ detail.cdrough }}</td>
00141:                         </tr>
00142:                     {% endif %}
00143:                 {% endfor %}
00144:                 </tbody>
00145:             </table>
00146:         </div>
00147:     </div>
00148:
00149:     <!-- Totals -->
00150:     <div class="row mb-2">
00151:         <div class="col-12 mb-2">
00152:             <strong>LIGHTING CIRCUITS:</strong> {{ lighting_circuits }}
00153:             <strong class="ms-3">OPTIONAL PAY: $</strong> {{ contract.total_options }}
00154:         </div>
00155:         <div class="col-12 text-end" style="font-size: 13px">
00156:             <span class="fw-bold">GRAND TOTAL:</span> $ {{ contract.total }} + _______
_____ = ____________
00157:         </div>
00158:     </div>
00159:     <!-- Sexto Row -->
00160:     <div class="row mb-3" style="font-size: 9px">
00161:         <div class="col-12">
00162:             <p class="fst-italic mb-1">Responsabilidades del Contratista</p>
00163:
00164:             <p class="text-justify mx-3 mb-1">El contratista es responsable de complet
ar el
00165:                 trabajo de
00166:                 electricidad asignado al 100%. Por lo tanto, debe
00167:                 asegurar que disponga de todo el material necesario y revisar todo el
trabajo realizado para
00168:                 entregar la
00169:                 casa con un "Hot Check". Además, es responsable de cualquier vehículo
que se le haya otorgado, así
00170:                 como de
00171:                 su propia seguridad mientras cumple con su deber de realizar el trabaj
o asignado por Division16
00172:                 LLC.</p>
```

ctrctsapp\templates\contract_pdf.html

```
00173:
00174:             <p class="text-justify mx-3 mb-2">El contratista deberá tomar todas las me
didas
00175:                 necesarias
00176:                 para garantizar que el trabajo se realice de manera
00177:                 segura y conforme a los estándares de calidad requeridos. Cualquier in
cumplimiento en estas
00178:                 responsabilidades será considerado como una violación de los términos
del contrato y puede resultar
00179:                 en
00180:                 sanciones correspondientes.</p>
00181:
00182:             <p class="fst-italic mb-1">Contractor Responsibilities</p>
00183:
00184:             <p class="text-justify mx-3 mb-1">The contractor is responsible for comple
ting the
00185:                 assigned electrical work 100%. Therefore, they must ensure
00186:                 that all necessary materials are available and review all completed wo
rk to deliver the house with a
00187:                 "Hot
00188:                 Check". Additionally, the contractor is responsible for any vehicle th
at has been provided to them,
00189:                 as well as their own safety while performing the duties assigned by Di
vision16 LLC.</p>
00190:
00191:             <p class="text-justify mx-3 mb-1">The contractor must take all necessary m
easures to
00192:                 ensure that the work is carried out safely and in
00193:                 accordance with the required quality standards. Any failure to meet th
ese responsibilities will be
00194:                 considered a breach of contract and may result in appropriate penaltie
s.</p>
00195:         </div>
00196:     </div>
00197:     <div class="row mt-1" style="font-size: 11px">
00198:         <div class="col-6 fst-italic mb-0">
00199:             CREW LEADER: _____________________________
00200:             <span class="ms-1">DATE: _____________</span>
00201:         </div>
00202:         <div class="col-6 fst-italic mb-0 text-end ">
00203:             SUPERVISOR: _____________________________
00204:             <span class="ms-1">DATE: _____________</span>
00205:         </div>
00206:     </div>
00207: </div>
00208: </body>
00209: </html>
```

ctrctsapp\templates\forgot_password_instructions.html

```
00001: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
00002: <html>
00003:   <head>
00004:     <!-- Compiled with Bootstrap Email version: 1.5.1 --><meta http-equiv="x-ua-compat
ible" content="ie=edge">
00005:     <meta name="x-apple-disable-message-reformatting">
00006:     <meta name="viewport" content="width=device-width, initial-scale=1">
00007:     <meta name="format-detection" content="telephone=no, date=no, address=no, email=no
">
00008:     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
00009:     <style type="text/css">
00010:       body,table,td{font-family:Helvetica,Arial,sans-serif !important}.ExternalClass{w
idth:100%}.ExternalClass,.ExternalClass p,.ExternalClass span,.ExternalClass font,.ExternalCl
ass td,.ExternalClass div{line-height:150%}a{text-decoration:none}*{color:inherit}a[x-apple-d
ata-detectors],u+#body a,#MessageViewBody a{color:inherit;text-decoration:none;font-size:inhe
rit;font-family:inherit;font-weight:inherit;line-height:inherit}img{-ms-interpolation-mode:bi
cubic}table:not([class^=s-]){font-family:Helvetica,Arial,sans-serif;mso-table-lspace:0pt;mso-
table-rspace:0pt;border-spacing:0px;border-collapse:collapse}table:not([class^=s-]) td{border
-spacing:0px;border-collapse:collapse}@media screen and (max-width: 600px){.w-full,.w-full>tb
ody>tr>td{width:100% !important}*[class*=s-lg-]>tbody>tr>td{font-size:0 !important;line-heigh
t:0 !important;height:0 !important}.s-2>tbody>tr>td{font-size:8px !important;line-height:8px
!important;height:8px !important}.s-5>tbody>tr>td{font-size:20px !important;line-height:20px
!important;height:20px !important}.s-10>tbody>tr>td{font-size:40px !important;line-height:40p
x !important;height:40px !important}}
00011:     </style>
00012:   </head>
00013:   <body class="bg-light" style="outline: 0; width: 100%; min-width: 100%; height: 100%
; -webkit-text-size-adjust: 100%; -ms-text-size-adjust: 100%; font-family: Helvetica, Arial,
sans-serif; line-height: 24px; font-weight: normal; font-size: 16px; -moz-box-sizing: border-
box; -webkit-box-sizing: border-box; box-sizing: border-box; color: #000000; margin: 0; paddi
ng: 0; border-width: 0;" bgcolor="#f7fafc">
00014:     <table class="bg-light body" valign="top" role="presentation" border="0" cellpaddi
ng="0" cellspacing="0" style="outline: 0; width: 100%; min-width: 100%; height: 100%; -webkit
-text-size-adjust: 100%; -ms-text-size-adjust: 100%; font-family: Helvetica, Arial, sans-seri
f; line-height: 24px; font-weight: normal; font-size: 16px; -moz-box-sizing: border-box; -web
kit-box-sizing: border-box; box-sizing: border-box; color: #000000; margin: 0; padding: 0; bo
rder-width: 0;" bgcolor="#f7fafc">
00015:       <tbody>
00016:         <tr>
00017:           <td valign="top" style="line-height: 24px; font-size: 16px; margin: 0;" alig
n="left" bgcolor="#f7fafc">
00018:             <table class="container" role="presentation" border="0" cellpadding="0" ce
llspacing="0" style="width: 100%;">
00019:               <tbody>
00020:                 <tr>
00021:                   <td align="center" style="line-height: 24px; font-size: 16px; margin
: 0; padding: 0 16px;">
00022:                     <!--[if (gte mso 9)|(IE)]>
00023:                       <table align="center" role="presentation">
00024:                         <tbody>
00025:                           <tr>
00026:                             <td width="600">
00027:                     <![endif]-->
00028:                     <table align="center" role="presentation" border="0" cellpadding="
0" cellspacing="0" style="width: 100%; max-width: 600px; margin: 0 auto;">
00029:                       <tbody>
00030:                         <tr>
00031:                           <td style="line-height: 24px; font-size: 16px; margin: 0;" a
lign="left">
```

ctrctsapp\templates\forgot_password_instructions.html

```
00032:                              <table class="s-10 w-full" role="presentation" border="0"
cellpadding="0" cellspacing="0" style="width: 100%;" width="100%">
00033:                                <tbody>
00034:                                  <tr>
00035:                                    <td style="line-height: 40px; font-size: 40px; width
: 100%; height: 40px; margin: 0;" align="left" width="100%" height="40">
00036:                                       
00037:                                    </td>
00038:                                  </tr>
00039:                                </tbody>
00040:                              </table>
00041:                              <table class="card" role="presentation" border="0" cellpad
ding="0" cellspacing="0" style="border-radius: 6px; border-collapse: separate !important; wid
th: 100%; overflow: hidden; border: 1px solid #e2e8f0;" bgcolor="#ffffff">
00042:                                <tbody>
00043:                                  <tr>
00044:                                    <td style="line-height: 24px; font-size: 16px; width
: 100%; margin: 0;" align="left" bgcolor="#ffffff">
00045:                                      <table class="card-body" role="presentation" borde
r="0" cellpadding="0" cellspacing="0" style="width: 100%;">
00046:                                        <tbody>
00047:                                          <tr>
00048:                                            <td style="line-height: 24px; font-size: 16p
x; width: 100%; margin: 0; padding: 20px;" align="left">
00049:                                              <h1 class="h3" style="padding-top: 0; padd
ing-bottom: 0; font-weight: 500; vertical-align: baseline; font-size: 28px; line-height: 33.6
px; margin: 0;" align="left">Reset password instructions</h1>
00050:                                              <table class="s-2 w-full" role="presentati
on" border="0" cellpadding="0" cellspacing="0" style="width: 100%;" width="100%">
00051:                                                <tbody>
00052:                                                  <tr>
00053:                                                    <td style="line-height: 8px; font-si
ze: 8px; width: 100%; height: 8px; margin: 0;" align="left" width="100%" height="8">
00054:                                                       
00055:                                                    </td>
00056:                                                  </tr>
00057:                                                </tbody>
00058:                                              </table>
00059:                                              <table class="s-5 w-full" role="presentati
on" border="0" cellpadding="0" cellspacing="0" style="width: 100%;" width="100%">
00060:                                                <tbody>
00061:                                                  <tr>
00062:                                                    <td style="line-height: 20px; font-s
ize: 20px; width: 100%; height: 20px; margin: 0;" align="left" width="100%" height="20">
00063:                                                       
00064:                                                    </td>
00065:                                                  </tr>
00066:                                                </tbody>
00067:                                              </table>
00068:                                              <table class="hr" role="presentation" bord
er="0" cellpadding="0" cellspacing="0" style="width: 100%;">
00069:                                                <tbody>
00070:                                                  <tr>
00071:                                                    <td style="line-height: 24px; font-s
ize: 16px; border-top-width: 1px; border-top-color: #e2e8f0; border-top-style: solid; height:
 1px; width: 100%; margin: 0;" align="left">
00072:                                                    </td>
00073:                                                  </tr>
00074:                                                </tbody>
00075:                                              </table>
```

ctrctsapp\templates\forgot_password_instructions.html

```
00076:                                                <table class="s-5 w-full" role="presentati
on" border="0" cellpadding="0" cellspacing="0" style="width: 100%;" width="100%">
00077:                                                  <tbody>
00078:                                                    <tr>
00079:                                                      <td style="line-height: 20px; font-s
ize: 20px; width: 100%; height: 20px; margin: 0;" align="left" width="100%" height="20">
00080:                                                         
00081:                                                      </td>
00082:                                                    </tr>
00083:                                                  </tbody>
00084:                                                </table>
00085:                                                <div class="space-y-3">
00086:                                                  <p class="text-gray-700" style="line-hei
ght: 24px; font-size: 16px; color: #4a5568; width: 100%; margin: 0;" align="left">
00087:                                                    Hello {{username}}!. <br>
00088:                                                    <br>
00089:                                                    Someone requested a link to change you
r password. Click the button below to proceed.
00090:                                                  </p>
00091:                                                </div>
00092:                                                <table class="s-5 w-full" role="presentati
on" border="0" cellpadding="0" cellspacing="0" style="width: 100%;" width="100%">
00093:                                                  <tbody>
00094:                                                    <tr>
00095:                                                      <td style="line-height: 20px; font-s
ize: 20px; width: 100%; height: 20px; margin: 0;" align="left" width="100%" height="20">
00096:                                                         
00097:                                                      </td>
00098:                                                    </tr>
00099:                                                  </tbody>
00100:                                                </table>
00101:                                                <table class="hr" role="presentation" bord
er="0" cellpadding="0" cellspacing="0" style="width: 100%;">
00102:                                                  <tbody>
00103:                                                    <tr>
00104:                                                      <td style="line-height: 24px; font-s
ize: 16px; border-top-width: 1px; border-top-color: #e2e8f0; border-top-style: solid; height:
 1px; width: 100%; margin: 0;" align="left">
00105:                                                      </td>
00106:                                                    </tr>
00107:                                                  </tbody>
00108:                                                </table>
00109:                                                <table class="s-5 w-full" role="presentati
on" border="0" cellpadding="0" cellspacing="0" style="width: 100%;" width="100%">
00110:                                                  <tbody>
00111:                                                    <tr>
00112:                                                      <td style="line-height: 20px; font-s
ize: 20px; width: 100%; height: 20px; margin: 0;" align="left" width="100%" height="20">
00113:                                                         
00114:                                                      </td>
00115:                                                    </tr>
00116:                                                  </tbody>
00117:                                                </table>
00118:                                                <table class="btn btn-primary" role="prese
ntation" border="0" cellpadding="0" cellspacing="0" style="border-radius: 6px; border-collaps
e: separate !important;">
00119:                                                  <tbody>
00120:                                                    <tr>
00121:                                                      <td style="line-height: 24px; font-s
ize: 16px; border-radius: 6px; margin: 0;" align="center" bgcolor="#0d6efd">
```

ctrctsapp\templates\forgot_password_instructions.html

```
00122:                                                    <a href="{{reset_url}}" target="_b
lank" style="color: #ffffff; font-size: 16px; font-family: Helvetica, Arial, sans-serif; text
-decoration: none; border-radius: 6px; line-height: 20px; display: block; font-weight: normal
; white-space: nowrap; background-color: #0d6efd; padding: 8px 12px; border: 1px solid #0d6ef
d;">Reset password</a>
00123:                                                  </td>
00124:                                                </tr>
00125:                                              </tbody>
00126:                                            </table>
00127:                                            <br>
00128:                                            <table class="s-5 w-full" role="presentati
on" border="0" cellpadding="0" cellspacing="0" style="width: 100%;" width="100%">
00129:                                              <tbody>
00130:                                                <tr>
00131:                                                  <td style="line-height: 20px; font-s
ize: 20px; width: 100%; height: 20px; margin: 0;" align="left" width="100%" height="20">
00132:                                                     
00133:                                                  </td>
00134:                                                </tr>
00135:                                              </tbody>
00136:                                            </table>
00137:                                            <p class="text-gray-700" style="line-heigh
t: 24px; font-size: 16px; color: #4a5568; width: 100%; margin: 0;" align="left">If you are un
able to click the button, please visit the following link
00138:                                              <br>
00139:                                              <br>
00140:                                              {{reset_url}}
00141:                                            </p>
00142:                                          </td>
00143:                                        </tr>
00144:                                      </tbody>
00145:                                    </table>
00146:                                  </td>
00147:                                </tr>
00148:                              </tbody>
00149:                            </table>
00150:                            <table class="s-10 w-full" role="presentation" border="0"
cellpadding="0" cellspacing="0" style="width: 100%;" width="100%">
00151:                              <tbody>
00152:                                <tr>
00153:                                  <td style="line-height: 40px; font-size: 40px; width
: 100%; height: 40px; margin: 0;" align="left" width="100%" height="40">
00154:                                     
00155:                                  </td>
00156:                                </tr>
00157:                              </tbody>
00158:                            </table>
00159:                          </td>
00160:                        </tr>
00161:                      </tbody>
00162:                    </table>
00163:                    <!--[if (gte mso 9)|(IE)]>
00164:                    </td>
00165:                  </tr>
00166:                </tbody>
00167:              </table>
00168:                    <![endif]-->
00169:                  </td>
00170:                </tr>
00171:              </tbody>
```

ctrctsapp\templates\forgot_password_instructions.html

```
00172:             </table>
00173:           </td>
00174:         </tr>
00175:       </tbody>
00176:     </table>
00177:   </body>
00178: </html>
```

ctrctsapp\templates\password_reset_confirm.html

```
00001: <!DOCTYPE html>
00002: <html lang="es">
00003: <head>
00004:     <meta charset="UTF-8">
00005:     <meta name="viewport" content="width=device-width, initial-scale=1.0">
00006:     <title>Restablecer Contraseña</title>
00007:     <link rel="stylesheet" href="{% static 'css/styles.css' %}">  <!-- Asegúrate de te
ner un archivo CSS si es necesario -->
00008: </head>
00009: <body>
00010:     <div class="container">
00011:         <h1>Restablecer Contraseña</h1>
00012:         <p>Por favor, ingresa tu nueva contraseña.</p>
00013:         <form method="post">
00014:             {% csrf_token %}
00015:             {{ form.as_p }}  <!-- Renderiza el formulario de cambio de contraseña -->
00016:             <button type="submit">Restablecer Contraseña</button>
00017:         </form>
00018:         <p>
00019:             <a href="{% url 'password_reset' %}">¿Olvidaste tu contraseña?</a>
00020:         </p>
00021:     </div>
00022: </body>
00023: </html>
```

ctrctsapp\templates\password_reset_email.html

```
00001: <p>Hola,</p>
00002: <p>Para restablecer su contraseña, haga clic en el siguiente enlace:</p>
00003: <a href="http://{{ domain }}/reset/{{ uid }}/{{ token }}/">Restablecer contraseña</a>
00004: <p>Si no solicitó este cambio, ignore este correo.</p>
```

ctrctsapp\tests.py

```
00001: from django.test import TestCase
00002:
00003: # Create your tests here.
```

ctrctsapp\urls.py

```
00001: from django.urls import path, include
00002: from rest_framework.routers import DefaultRouter
00003: from .views import (
00004:     ContractViewSet, WorkPriceViewSet, ContractDetailsViewSet,
00005:     BuilderViewSet, JobViewSet, HouseModelViewSet, LoginView,
00006:     validate_token, logout_view, weekly_summary, weekly_summary_list,
00007:     monthly_summary, login_view, user_permissions, request_password_reset, reset_passw
ord_confirm,
00008:     UserDetailView, get_house_model_jobs, geocode_view, download_contract_pdf,
00009:     BuilderReadOnlyViewSet, JobReadOnlyViewSet, HouseReadOnlyViewSet
00010: )
00011:
00012: router = DefaultRouter()
00013: router.register('contract', ContractViewSet, basename='contract')
00014: router.register('workprice', WorkPriceViewSet, basename='workprice')
00015: router.register('contract_details', ContractDetailsViewSet, basename='contract_details
')
00016: router.register('builder', BuilderViewSet, basename='builder')
00017: router.register('job', JobViewSet, basename='job')
00018: router.register('house_model', HouseModelViewSet, basename='house_model')
00019: router.register('api/builders', BuilderReadOnlyViewSet, basename='api-builders')
00020: router.register('api/jobs', JobReadOnlyViewSet, basename='api-jobs')
00021: router.register('api/houses', HouseReadOnlyViewSet, basename='api-houses')
00022:
00023: urlpatterns = [
00024:     path('api/', include(router.urls)),
00025:     path('api/jobs_by_builder/', JobViewSet.as_view({'get': 'jobs_by_builder'}), name=
'jobs-by-builder'),
00026:     path('api/house_models_by_job/', HouseModelViewSet.as_view({'get': 'house_models_b
y_job'}), name='house-models-by-job'),
00027:     path('api/house_models/<int:house_model_id>/jobs/', get_house_model_jobs, name='ge
t_house_model_jobs'), # http://localhost:8000/api/house_models/55/jobs/
00028:     path('api/login/', LoginView.as_view(), name='login'),  # Usar la vista personaliz
ada
00029:     path('api/validate-token/', validate_token, name='validate-token'),
00030:     path('api/logout/', logout_view, name='logout'),
00031:     path('api/login/', login_view, name='api_login'),
00032:     path('api/user-permissions/', user_permissions, name='user_permissions'),
00033:     path('api/request-password-reset/', request_password_reset, name='request_password
_reset'),
00034:     path('api/password-reset-confirm/<uidb64>/<token>/', reset_password_confirm, name=
'password_reset_confirm'),
00035:     path('api/weekly_summary/', weekly_summary, name='weekly_summary'),
00036:     path('api/weekly_summary_list/', weekly_summary_list, name='weekly_summary_list'),
00037:     path('api/monthly_summary/', monthly_summary, name='monthly_summary'),
00038:     path('api/user_detail/', UserDetailView.as_view(), name='user_detail'),
00039:     path('api/geocode/', geocode_view, name='geocode'),
00040:     path('web/contract-pdf/<int:contract_id>', download_contract_pdf, name='contract-p
df'),
00041:     path('api/builder/<int:pk>/workprices/', BuilderViewSet.as_view({'get': 'workprice
s'}), name='builder-workprices'),
00042:     path('api/builder/<int:pk>/assign-workprices/', BuilderViewSet.as_view({'post': 'a
ssign_workprices'}), name='assign-workprices'),
00043:
00044: ] + router.urls
```

ctrctsapp\utils.py

```
00001: import requests
00002:
00003: def geocode_address(address):
00004:     """
00005:     Geocodifica una dirección utilizando la API de Nominatim de OpenStreetMap.
00006:     :param address: Dirección a geocodificar (cadena de texto).
00007:     :return: Una tupla (latitud, longitud) si se encuentra la dirección; de lo contrar
io, (None, None).
00008:     """
00009:     url = "https://nominatim.openstreetmap.org/search"
00010:     params = {
00011:         "q": address,        # Dirección a buscar
00012:         "format": "json",    # Formato de la respuesta (JSON)
00013:         "addressdetails": 1, # Incluir detalles de la dirección
00014:     }
00015:
00016:     # Realizar la solicitud HTTP a la API
00017:     try:
00018:         response = requests.get(url, params=params, headers={"User-Agent": "django-app
"})
00019:         if response.status_code == 200:
00020:             results = response.json()
00021:             if results:
00022:                 # Extraer latitud y longitud del primer resultado
00023:                 location = results[0]
00024:                 return float(location["lat"]), float(location["lon"])
00025:     except Exception as e:
00026:         print(f"Error al geocodificar la dirección '{address}': {e}")
00027:
00028:     return None, None
```

ctrctsapp\views.py

```
00001: import base64
00002: import logging
00003: import json
00004: import math
00005: from decimal import Decimal
00006:
00007: from rest_framework.views import APIView # Importa la clase APIView de Django Rest Fra
mework, que es la base para crear vistas basadas en clases para APIs
00008: from rest_framework import viewsets, status
00009: from rest_framework.decorators import action, api_view, permission_classes
00010: from rest_framework.response import Response
00011: from django.http import JsonResponse
00012: from rest_framework.authtoken.models import Token
00013: from django.contrib.auth import authenticate
00014: from rest_framework.permissions import IsAuthenticated, DjangoModelPermissions, AllowA
ny, IsAuthenticatedOrReadOnly
00015: from rest_framework.authentication import TokenAuthentication
00016: from .models import Contract, WorkPrice, ContractDetails, Builder, Job, HouseModel
00017: from .serializers import (
00018:     ContractSerializer, WorkPriceSerializer, ContractDetailsSerializer,
00019:     BuilderSerializer, JobSerializer, HouseModelSerializer,
00020:     JobFilteredByBuilderSerializer, HouseModelFilteredByJobSerializer,
00021:     BuilderListSerializer, JobListSerializer, HouseListSerializer, ContractListSeriali
zer
00022: )
00023: from django.db.models import Sum, Count
00024: from django.db.models.functions import TruncWeek, TruncMonth
00025: from django.utils import timezone
00026: from django.conf import settings
00027: from datetime import datetime, timedelta
00028: import logging
00029:
00030: from django.contrib.auth.models import User
00031: from django.core.mail import send_mail
00032: from django.core.mail import EmailMultiAlternatives
00033: from django.utils.http import urlsafe_base64_encode, urlsafe_base64_decode
00034: from django.utils.encoding import force_bytes
00035: from django.contrib.auth.tokens import default_token_generator
00036: from django.shortcuts import get_object_or_404
00037: from django.http import JsonResponse, HttpResponse
00038: from django.template.loader import render_to_string
00039: from .utils import geocode_address
00040: from utils.datatable import handle_datatable_query
00041:
00042: from weasyprint import (
00043:     CSS,
00044:     HTML,
00045: )
00046: from weasyprint.text.fonts import FontConfiguration
00047:
00048: logger = logging.getLogger(__name__)
00049:
00050:
00051: def geocode_view(request):
00052:     """
00053:     Vista para geocodificar una dirección y devolver latitud y longitud.
00054:     """
00055:     address = request.GET.get(
00056:         "address")  # La dirección debe ser pasada como parámetro GET
00057:     if not address:
```

ctrctsapp\views.py

```
00058:         return JsonResponse({"error": "Address parameter is required"}, status=400)
00059:
00060:     latitude, longitude = geocode_address(address)
00061:     if latitude and longitude:
00062:         return JsonResponse({"address": address, "latitude": latitude, "longitude": lo
ngitude})
00063:     else:
00064:         return JsonResponse({"error": "Could not geocode address"}, status=404)
00065:
00066:
00067: @permission_classes([AllowAny])
00068: class UserDetailView(APIView):
00069:     def get(self, request):
00070:         user = request.user
00071:         return Response({
00072:             'id': user.id,
00073:             'username': user.username,
00074:         })
00075:
00076:
00077: @api_view(['POST'])
00078: @permission_classes([AllowAny])
00079: def request_password_reset(request):
00080:     if request.method == 'POST':
00081:         data = json.loads(request.body)
00082:         email = data.get('email')
00083:         try:
00084:             user = User.objects.get(email=email)
00085:             token = default_token_generator.make_token(user)
00086:             uid = urlsafe_base64_encode(force_bytes(user.pk))
00087:             reset_url = f"{settings.FRONT_URL}/reset-password-confirm?uid={uid}&token=
{token}"
00088:             text = f'Hello {user.username}! \n\n\nSomeone requested a link to change y
our password. Click the link below to proceed. \n\n\n{reset_url}'
00089:             html_content = render_to_string(
00090:                 "forgot_password_instructions.html",
00091:                 context={"username": user.username, 'reset_url': reset_url},
00092:             )
00093:             msg = EmailMultiAlternatives(
00094:                 'Reset password instructions',
00095:                 text,
00096:                 settings.DEFAULT_FROM_EMAIL,
00097:                 [email],
00098:             )
00099:             msg.attach_alternative(html_content, "text/html")
00100:             msg.send()
00101:             return JsonResponse({'message': 'Email sent'}, status=200)
00102:         except User.DoesNotExist:
00103:             return JsonResponse({'error': 'Error for recovery password'}, status=404)
00104:     return JsonResponse({'error': 'Method not allowed'}, status=405)
00105:
00106:
00107: @api_view(['POST'])
00108: @permission_classes([AllowAny])
00109: def reset_password_confirm(request, uidb64, token):
00110:     if request.method == 'POST':
00111:         data = json.loads(request.body)
00112:         new_password = data.get('new_password')
00113:
00114:         try:
```

ctrctsapp\views.py

```
00115:             uid = urlsafe_base64_decode(uidb64).decode()
00116:             user = get_object_or_404(User, pk=uid)
00117:
00118:             if default_token_generator.check_token(user, token):
00119:                 user.set_password(new_password)
00120:                 user.save()
00121:                 return JsonResponse({'message': 'Password reset successfully'}, status
=200)
00122:             else:
00123:                 return JsonResponse({'error': 'Invalid token'}, status=400)
00124:         except Exception as e:
00125:             return JsonResponse({'error': str(e)}, status=400)
00126:
00127:     return JsonResponse({'error': 'Method not allowed'}, status=405)
00128:
00129:
00130: @api_view(['GET'])
00131: @permission_classes([IsAuthenticated])
00132: def user_permissions(request):
00133:     permissions = list(request.user.get_all_permissions())
00134:     # print(f"Permissions in View>>: {permissions}")
00135:     return Response({'permissions': permissions})
00136:
00137: @api_view(['GET'])
00138: def validate_token(request):
00139:     token_key = request.headers.get('Authorization')
00140:     if token_key:
00141:         # Obtener solo el valor del token
00142:         token_key = token_key.replace('Token ', '')
00143:         try:
00144:             token = Token.objects.get(key=token_key)
00145:             # print(f"Token found in View>>: {token}")
00146:             return Response({'valid': True}, status=status.HTTP_200_OK)
00147:         except Token.DoesNotExist:
00148:             return Response({'valid': False}, status=status.HTTP_401_UNAUTHORIZED)
00149:     return Response({'valid': False}, status=status.HTTP_401_UNAUTHORIZED)
00150:
00151: class LoginView(APIView):
00152:     permission_classes = [AllowAny]  # Permite acceso sin autenticación
00153:
00154:     def post(self, request):
00155:         username = request.data.get('username')
00156:         password = request.data.get('password')
00157:         # print(f"Intentando LoginView: {username} {password}")
00158:         user = authenticate(username=username, password=password)
00159:
00160:         if user is not None:
00161:             token, created = Token.objects.get_or_create(user=user)
00162:             permissions = list(user.get_all_permissions())
00163:             return Response({
00164:                 'token': token.key,
00165:                 'permissions': permissions
00166:             }, status=status.HTTP_200_OK)
00167:         else:
00168:             return Response({'error': 'Credenciales inválidas'}, status=status.HTTP_40
1_UNAUTHORIZED)
00169:
00170: @api_view(['POST'])
00171: @permission_classes([AllowAny])
00172: def login_view(request):
```

ctrctsapp\views.py

```
00173:     # print("Request data:", request.data)
00174:     username = request.data.get('username')
00175:     password = request.data.get('password')
00176:     # print(f"Intentando login_view: {username} {password}")
00177:     user = authenticate(username=username, password=password)
00178:     if user:
00179:         token, _ = Token.objects.get_or_create(user=user)
00180:         return Response({'token': token.key}, status=status.HTTP_200_OK)
00181:     else:
00182:         return Response({'error': 'Invalid credentials'}, status=status.HTTP_401_UNAUT
HORIZED)
00183:
00184:
00185: @api_view(['POST'])
00186: @permission_classes([AllowAny])
00187: def logout_view(request):
00188:     try:
00189:         token = request.auth
00190:         token.delete()  # Eliminar el token del usuario actual
00191:         response = Response({"message": "Session closed successfully."}, status=200)
00192:         response.delete_cookie('userPermissions')  # Eliminar la cookie de permisos
00193:
00194:         # Agregar instrucciones para eliminar datos de sessionStorage
00195:         response['X-Delete-Session-Storage'] = 'authToken,userPermissions'
00196:
00197:         return response
00198:     except AttributeError:
00199:         return Response({"error": "No token found."}, status=400)
00200:
00201:
00202: @api_view(['GET'])
00203: # @permission_classes([IsAuthenticated])
00204: def monthly_summary(request): # BarChart.vue component
00205:     # Calcular la fecha de hace 12 meses desde hoy
00206:     twelve_months_ago = timezone.now() - timezone.timedelta(days=365)
00207:     # Consulta que agrupa los contratos por mes y tipo de trabajo (trim o rough)
00208:     summary = (
00209:         Contract.objects
00210:         .filter(doc_type="Contract", date_created__gte=twelve_months_ago) # solo contr
atos de los últimos 12 meses
00211:         .annotate(month=TruncMonth('date_created'))  # Agrupar por mes
00212:         .values('month', 'type')  # Seleccionar mes y tipo de trabajo
00213:         .annotate(total_contracts=Count('id'))  # Contar contratos por grupo
00214:         .order_by('month', 'type')  # Ordenar por mes y tipo
00215:     )
00216:
00217:     # Formatear los datos para devolver en JSON
00218:     result = [
00219:         {
00220:             'month': entry['month'].strftime('%Y-%m'),  # Formato 'YYYY-MM'
00221:             'job_type': entry['type'],  # Tipo de trabajo (trim o rough)
00222:             'total_contracts': entry['total_contracts']  # Total de contratos en ese m
es/tipo
00223:         }
00224:         for entry in summary
00225:     ]
00226:
00227:     return Response(result)
00228:
00229:
```

ctrctsapp\views.py

```
00230: @api_view(['GET'])
00231: @permission_classes([IsAuthenticated])
00232: def weekly_summary_list(request): # WeeklySummaryListComponent.vue component
00233:     # Obtener fechas desde los parámetros de la solicitud
00234:     start_date_str = request.GET.get('start_date', '2024-07-01')  # Default si no se p
roporciona
00235:     end_date_str = request.GET.get('end_date', '2025-08-31')      # Default si no se p
roporciona
00236:
00237:     # Convertir a objetos datetime
00238:     start_date = datetime.strptime(start_date_str, '%Y-%m-%d')
00239:     # end_date = datetime.strptime(end_date_str, '%Y-%m-%d')
00240:
00241:     contracts = Contract.objects.filter(
00242:         date_created__gte=start_date, doc_type="Contract"
00243:     ).annotate(week=TruncWeek('date_created')).values(
00244:         'week', 'job__name', 'type'
00245:     ).annotate(total_contracts=Count('id')).order_by('week', 'job__name', 'type')
00246:
00247:     result = []
00248:     for contract in contracts:
00249:         week_start = contract['week']
00250:         week_end = week_start + timedelta(days=6)
00251:         result.append({
00252:             'start_of_week': week_start,
00253:             'end_of_week': week_end,
00254:             'job__name': contract['job__name'],
00255:             'total_contracts': contract['total_contracts'],
00256:         })
00257:
00258:     return JsonResponse(result, safe=False)
00259:
00260:
00261: @api_view(['GET'])
00262: # @permission_classes([IsAuthenticated])
00263: def weekly_summary(request): # AreaChart.vue component
00264:     # Calcular la fecha de hace 52 semanas desde hoy
00265:     one_year_ago = timezone.now() - timezone.timedelta(weeks=52)
00266:
00267:     # Consulta que agrupa por semana y tipo de contrato (Trim o Rough)
00268:     data = (
00269:         Contract.objects
00270:         .filter(doc_type="Contract")
00271:         .annotate(week=TruncWeek('date_created'))
00272:         .values('week', 'type')
00273:         .annotate(total=Sum('total'), total_contracts=Count('id'))
00274:         .order_by('week', 'type')
00275:     )
00276:
00277:     # Formatear las fechas de la semana
00278:     formatted_data = [
00279:         {
00280:             "week": entry["week"].strftime('%m-%d-%Y'),  # Formatear la fecha
00281:             "type": entry["type"],
00282:             "total": entry["total"],
00283:             "total_contracts": entry["total_contracts"],
00284:         }
00285:         for entry in data
00286:     ]
00287:
```

ctrctsapp\views.py

```
00288:     return Response(formatted_data)
00289:
00290:
00291: class ContractViewSet(viewsets.ModelViewSet):
00292:     queryset = Contract.objects.all()
00293:     serializer_class = ContractSerializer
00294:     authentication_classes = [TokenAuthentication]
00295:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00296:
00297:     def create(self, request, *args, **kwargs):
00298:         # Log the incoming request data
00299:         # print("Request data:", request.data)
00300:
00301:         serializer = self.get_serializer(data=request.data)
00302:         if serializer.is_valid():
00303:             self.perform_create(serializer)
00304:             headers = self.get_success_headers(serializer.data)
00305:             return Response(serializer.data, status=status.HTTP_201_CREATED, headers=h
eaders)
00306:
00307:         # Log the serializer errors
00308:         logger.error("Serializer errors: %s", serializer.errors)
00309:         return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
00310:
00311:     def update(self, request, *args, **kwargs):
00312:         partial = kwargs.pop('partial', False)
00313:         instance = self.get_object()
00314:         serializer = self.get_serializer(
00315:             instance, data=request.data, partial=partial)
00316:
00317:         # Log the incoming request data
00318:         #print("Request data (update):", request.data)
00319:
00320:         if serializer.is_valid():
00321:             self.perform_update(serializer)
00322:             return Response(serializer.data)
00323:
00324:         # Log the serializer errors
00325:         logger.error("Serializer errors: %s", serializer.errors)
00326:         return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
00327:
00328:     def to_internal_value(self, data):
00329:         if 'lot' in data and (data['lot'] is None or (data['lot'] == '' or data['lot']
 == "null")):
00330:             data['lot'] = None
00331:         return super().to_internal_value(data)
00332:
00333:     @action(detail=False, methods=['get'], url_path='validate-lot')
00334:     def validate_contract(self, request):
00335:         lot = request.GET.get('lot')
00336:         type_ = request.GET.get('type')
00337:         job_id = request.GET.get('job')
00338:         address = request.GET.get('address')
00339:
00340:         if not type_:
00341:             return JsonResponse({'error': 'Type is a required parameter.'}, status=400
)
00342:
00343:         # Validar por lote
00344:         if lot:
```

ctrctsapp\views.py

```
00345:             if job_id == 'S/L':
00346:                 lot_exists = Contract.objects.filter(
00347:                     lot=lot, type=type_, address=address).exists()
00348:             else:
00349:                 lot_exists = Contract.objects.filter(
00350:                     lot=lot, type=type_, job_id=job_id).exists()
00351:             if lot_exists:
00352:                 return JsonResponse({'exists': True})
00353:
00354:         # Validar por dirección cuando no hay lote
00355:         if address and (not lot or lot == ''):
00356:             address_exists = Contract.objects.filter(
00357:                 address=address, type=type_).exists()
00358:             if address_exists:
00359:                 return JsonResponse({'exists': True})
00360:
00361:         return JsonResponse({'exists': False})
00362:
00363:     @action(detail=False, methods=['get'], url_path='user-contracts')
00364:     def get_user_contracts(self, request):
00365:         user = request.user
00366:         # Check if the user is in any crew
00367:         user_jobs = Job.objects.filter(crews__members=user)
00368:         # If the user is part of a crew, filter contracts by user's jobs
00369:         if user_jobs.exists():
00370:             contracts = Contract.objects.filter(job__in=user_jobs)
00371:         else:
00372:             # If the user is not in any crew, return all contracts
00373:             contracts = Contract.objects.all()
00374:         serializer = self.get_serializer(contracts, many=True)
00375:         return Response(serializer.data)
00376:
00377:
00378:     def perform_update(self, serializer):
00379:         # Obtener el objeto original (instancia del contrato)
00380:         instance = self.get_object()
00381:
00382:         # Lista de campos relevantes que deben ser monitoreados para cambios
00383:         relevant_fields = ['total', 'total_options', 'comment', 'lot', 'address']
00384:
00385:         # Inicializamos la bandera needs_reprint en False
00386:         needs_reprint = instance.needs_reprint  # Mantener el valor actual si no hay c
ambios
00387:
00388:         # Iteramos sobre los campos relevantes para verificar si alguno ha cambiado
00389:         for field in relevant_fields:
00390:             # Obtener el nuevo valor (de la solicitud)
00391:             new_value = self.request.data.get(field, None)
00392:             # Obtener el valor actual del campo en la base de datos
00393:             old_value = getattr(instance, field, None)
00394:
00395:             # Compara valores de tipo str
00396:             if isinstance(new_value, str) and isinstance(old_value, str):
00397:                 if new_value.strip() != old_value.strip():
00398:                     needs_reprint = True  # Marcar como modificado si hay diferencia
00399:                     break  # Si encontramos una diferencia, no es necesario seguir ver
ificando
00400:
00401:             # Compara valores de tipo decimal (por ejemplo, 'total' o 'total_options')
00402:             elif isinstance(new_value, str) and isinstance(old_value, Decimal):
```

ctrctsapp\views.py

```
00403:                 try:
00404:                     new_value = Decimal(new_value)  # Convertir el nuevo valor a Decim
al
00405:                     if new_value != old_value:
00406:                         needs_reprint = True  # Marcar como modificado si hay diferenc
ia
00407:                         break
00408:                 except ValueError:
00409:                     pass  # En caso de que no se pueda convertir, no cambiamos needs_r
eprint
00410:
00411:             # Compara otros casos en los que los valores deben ser tratados como str (
por ejemplo, 'lot', 'address')
00412:             elif isinstance(new_value, str) and isinstance(old_value, str):
00413:                 if new_value.strip() != old_value.strip():
00414:                     needs_reprint = True  # Marcar como modificado si hay diferencia
00415:                     break
00416:
00417:         # ■ **Modificación clave:** No marcar needs_reprint si es un "Bid"
00418:         if instance.doc_type == 'Bid':
00419:             needs_reprint = False
00420:
00421:         # Guardamos el contrato con el valor actualizado de 'needs_reprint'
00422:         serializer.save(needs_reprint=needs_reprint)
00423:
00424:         # Realizamos la actualización del contrato
00425:         super().perform_update(serializer)
00426:
00427:
00428:     @action(detail=True, methods=['put'], url_path='mark-printed')
00429:     def mark_as_printed(self, request, pk=None):
00430:         # Obtener el contrato con el ID proporcionado en la URL
00431:         contract = self.get_object()
00432:
00433:         # Actualizar el campo 'needs_reprint' a False
00434:         contract.needs_reprint = False
00435:         contract.save()  # Guardar los cambios en la base de datos
00436:
00437:         return Response({'status': 'Contract marked as printed'})
00438:
00439:     # Remplaza datatable_contracts sin Lazy Load
00440:     # Lazy Load para DataTable de Contratos con filtros por supervisor
00441:     @action(detail=False, methods=['get'], url_path='datatable-contracts', permission_
classes=[AllowAny])
00442:     def datatable_contracts(self, request):
00443:         """
00444:         Lazy load para DataTables filtrado por comunidades (jobs) del usuario.
00445:         a) Check if the user is in any crew.
00446:         b) If the user is part of a crew, filter contracts by user's jobs.
00447:         c) If el usuario no pertenece a crews, retorna todos los contratos.
00448:         """
00449:         user_id = request.GET.get('user_id')
00450:         if not user_id:
00451:             return Response({'error': 'user_id is required'}, status=400)
00452:
00453:         try:
00454:             user = User.objects.get(id=user_id)
00455:         except User.DoesNotExist:
00456:             return Response({'error': 'User not found'}, status=404)
00457:
```

ctrctsapp\views.py

```
00458:         user_jobs = Job.objects.filter(crews__members=user)
00459:         if user_jobs.exists():
00460:             queryset = Contract.objects.filter(job__in=user_jobs)
00461:         else:
00462:             queryset = Contract.objects.all()
00463:
00464:         queryset = queryset.select_related('builder', 'job', 'house_model')
00465:         search_fields = [
00466:                             'id',
00467:                             'doc_type',
00468:                             'type',
00469:                             'date_created',
00470:                             'builder__name',
00471:                             'job__name',
00472:                             'house_model__name',
00473:                             'lot',
00474:                             'address',
00475:                             'sqft',
00476:                             'job_price',
00477:                             'total_options',
00478:                             'total',
00479:                         ]
00480:         return handle_datatable_query(request, queryset, ContractListSerializer, searc
h_fields)
00481:
00482: class ContractDetailsViewSet(viewsets.ModelViewSet):
00483:     queryset = ContractDetails.objects.all()
00484:     serializer_class = ContractDetailsSerializer
00485:     authentication_classes = [TokenAuthentication]
00486:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00487:
00488:
00489: class WorkPriceViewSet(viewsets.ModelViewSet):
00490:     queryset = WorkPrice.objects.all()
00491:     serializer_class = WorkPriceSerializer
00492:     authentication_classes = [TokenAuthentication]
00493:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00494:
00495:     def get_queryset(self):
00496:         queryset = super().get_queryset()
00497:         builder_id = self.request.query_params.get('builder', None)
00498:         if builder_id:
00499:             queryset = queryset.filter(builders__id=builder_id)
00500:         return queryset
00501:
00502:
00503: class BuilderViewSet(viewsets.ModelViewSet):
00504:     queryset = Builder.objects.all()
00505:     serializer_class = BuilderSerializer
00506:     authentication_classes = [TokenAuthentication]
00507:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00508:
00509:     @action(detail=True, methods=["get"])
00510:     def workprices(self, request, pk=None):
00511:         """Get Work Prices assigned to a specific Builder"""
00512:         builder = self.get_object()
00513:         work_prices = builder.work_prices.all()
00514:         serializer = WorkPriceSerializer(work_prices, many=True)
00515:         return Response(serializer.data)
00516:
```

ctrctsapp\views.py

```
00517:     @action(detail=True, methods=["post"])
00518:     def assign_workprices(self, request, pk=None):
00519:         """Assign Work Prices to a Builder"""
00520:         builder = self.get_object()
00521:         work_price_ids = request.data.get("work_price_ids", [])
00522:
00523:         # Assign selected Work Prices
00524:         builder.work_prices.set(WorkPrice.objects.filter(id__in=work_price_ids))
00525:
00526:         return Response({"message": "Assignments updated successfully"}, status=status
.HTTP_200_OK)
00527:
00528:
00529: class JobViewSet(viewsets.ModelViewSet):
00530:     queryset = Job.objects.all().select_related("builder").prefetch_related("crews")
00531:     serializer_class = JobSerializer
00532:     authentication_classes = [TokenAuthentication]
00533:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00534:
00535:     @action(detail=False, methods=['get'])
00536:     def jobs_by_builder(self, request):
00537:         builder_id = request.query_params.get('builder_id')
00538:         if builder_id:
00539:             jobs = Job.objects.filter(builder_id=builder_id)
00540:         else:
00541:             jobs = Job.objects.none()
00542:         serializer = JobFilteredByBuilderSerializer(jobs, many=True)
00543:         return Response(serializer.data)
00544:
00545:
00546: class HouseModelViewSet(viewsets.ModelViewSet):
00547:     queryset = HouseModel.objects.all()
00548:     serializer_class = HouseModelSerializer
00549:     authentication_classes = [TokenAuthentication]
00550:     permission_classes = [IsAuthenticated, DjangoModelPermissions]
00551:
00552:     @action(detail=False, methods=['get'])
00553:     def house_models_by_job(self, request):
00554:         job_id = request.query_params.get('job_id')
00555:         if job_id:
00556:             house_models = HouseModel.objects.filter(jobs__id=job_id)
00557:         else:
00558:             house_models = HouseModel.objects.none()
00559:         serializer = HouseModelFilteredByJobSerializer(house_models, many=True)
00560:         return Response(serializer.data)
00561:
00562: @api_view(['GET'])
00563: @permission_classes([AllowAny])
00564: def get_house_model_jobs(request, house_model_id):
00565:     try:
00566:         house_model = HouseModel.objects.get(id=house_model_id)
00567:         jobs = house_model.jobs.all()  # Get all jobs related to the house model
00568:
00569:         # Prepare the data to return
00570:         job_data = [
00571:             {
00572:                 "id": job.id,
00573:                 "name": job.name,
00574:                 "builder_id": job.builder.id,
00575:                 "builder_name": job.builder.name
```

ctrctsapp\views.py

```
00576:             }
00577:             for job in jobs
00578:         ]
00579:
00580:         return JsonResponse({"houseModel": {"id": house_model.id, "name": house_model.
name}, "jobs": job_data})
00581:     except HouseModel.DoesNotExist:
00582:         return JsonResponse({"error": "House Model not found"}, status=404)
00583:
00584:
00585: def lighting_circuits(sqft):
00586:   if sqft == 0:
00587:     return 0
00588:   num = sqft * 3 / 120 / 15
00589:   return math.ceil(num)  # Siempre redondea hacia arriba
00590:
00591: @api_view(['GET'])
00592: def download_contract_pdf(request, contract_id):
00593:     try:
00594:         contract = Contract.objects.get(id=contract_id)
00595:
00596:         if contract.type == "Trim":
00597:             details = contract.contract_details.filter(cdtrim__gt=0)
00598:         else:  # If type is "Rough"
00599:             details = contract.contract_details.filter(cdrough__gt=0)
00600:
00601:         mid_index = details.count() // 2  # Integer division
00602:         left_details = details[:mid_index]
00603:         right_details = details[mid_index:]
00604:
00605:         domain = request.get_host()
00606:         if 'phoenixelectricandair' in domain:
00607:             tenant_logo = 'media/tenant_logos/Logo-phoenix-w.png'
00608:         elif '192.168.0.248:8000' in domain or 'division16llc' in domain:
00609:             tenant_logo = 'media/tenant_logos/Logo-division-w.png'
00610:         else:
00611:             tenant_logo = 'media/tenant_logos/default-logo.png'
00612:
00613:         logo_url = request.build_absolute_uri('/' + tenant_logo)
00614:
00615:         # Prepare the data to return
00616:         context = {
00617:             'contract': contract,
00618:             'left_details': left_details,
00619:             'right_details': right_details,
00620:             'lighting_circuits': lighting_circuits(contract.sqft),
00621:             'logo_url': logo_url,
00622:         }
00623:         font_config = FontConfiguration()
00624:
00625:         html = render_to_string('contract_pdf.html', context)
00626:
00627:         pdf_file = HTML(string=html).write_pdf(font_config=font_config)
00628:         response = {
00629:             'file': base64.b64encode(pdf_file),
00630:             'filename': f'contract_{contract.pk}.pdf',
00631:             'file_type': 'application/pdf'
00632:         }
00633:         return Response(response, status=status.HTTP_200_OK)
00634:     except HouseModel.DoesNotExist:
```

ctrctsapp\views.py

```
00635:         return JsonResponse({"error": "House Model not found"}, status=404)
00636:
00637:
00638: class BuilderReadOnlyViewSet(viewsets.ReadOnlyModelViewSet):
00639:     """
00640:     This viewset automatically provides `list` and `retrieve` actions.
00641:     """
00642:     queryset = Builder.objects.all()
00643:     serializer_class = BuilderListSerializer
00644:     permission_classes = [IsAuthenticatedOrReadOnly]
00645:
00646:
00647: class JobReadOnlyViewSet(viewsets.ReadOnlyModelViewSet):
00648:     serializer_class = JobListSerializer
00649:     permission_classes = [IsAuthenticatedOrReadOnly]
00650:
00651:     def get_queryset(self):
00652:         qs = Job.objects.all().select_related('builder')
00653:         builder_id = self.request.query_params.get('builder_id')
00654:         q = self.request.query_params.get('q')
00655:
00656:         if builder_id:
00657:             qs = qs.filter(builder_id=builder_id)
00658:
00659:         if q:
00660:             qs = qs.filter(name__icontains=q)
00661:
00662:         return qs.order_by('name')
00663:
00664:
00665: class HouseReadOnlyViewSet(viewsets.ReadOnlyModelViewSet):
00666:     """
00667:     This viewset automatically provides `list` and `retrieve` actions.
00668:     """
00669:     queryset = HouseModel.objects.all()
00670:     serializer_class = HouseListSerializer
00671:     permission_classes = [IsAuthenticatedOrReadOnly]
```